AFIT/GOR/ENS/94M-07



# A MEAN VALUE ANALYSIS HEURISTIC FOR ANALYSIS OF AIRCRAFT SORTIE GENERATION

THESIS

Richard Carl Jenkins
Captain, USAF

AFIT/GOR/ENS/94M-07



Approved for public release; distribution unlimited

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | March 1994 | Master's Thesis |

**4. TITLE AND SUBTITLE**
A MEAN VALUE ANALYSIS HEURISTIC FOR ANALYSIS OF AIRCRAFT SORTIE GENERATION

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
Richard Carl Jenkins, Captain, USAF

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583

**8. PERFORMING ORGANIZATION REPORT NUMBER**
AFIT/GOR/ENS/94M-07

**9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSORING MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** (Maximum 200 words)

The primary objective of this study was to develop an analytical methodology based on the Mean Value Analysis algorithm that approximates the performance characteristics of a queueing network model (QNM) containing a fork-join queue with probabilistic branching. These performance characteristics are response time, throughput and queue length at each station. The QNM solved contains the essential features of the aircraft sortie generation process. The sensitivity of the method's accuracy to increases in server utilization was determined. The comparisons of the results of the MVA heuristic to the outputs of the Logistics Composite Model (LCOM) simulation indicate that the heuristic's accuracy decreased as server utilization increases. When server utilization was kept in realistic ranges, the results of the heuristic for a single fork-join queue were very accurate. For non-maintenance stations, results were within 1 to 2 percent of the LCOM simulation output. For stations on the fork-join queue paths, heuristic results were within 5 percent of the LCOM simulation's output for that portion of the network.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Queueing Theory, Queueing Network Models, Fork-Join Queue<br>Aircraft Sortie Generation, Maintenance Manpower | 88 |
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

AFIT/GOR/ENS/94M-07

# A MEAN VALUE ANALYSIS HEURISTIC FOR ANALYSIS OF AIRCRAFT SORTIE GENERATION

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Richard Carl Jenkins, B.S.
Captain, USAF

March, 1994

THESIS APPROVAL

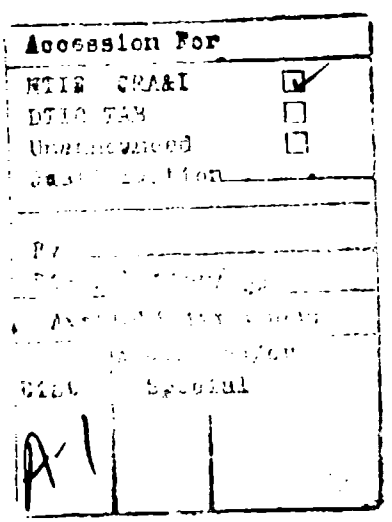
STUDENT: Richard C. Jenkins, Captain, USAF CLASS: GOR-94M

THESIS TITLE: A MEAN VALUE ANALYSIS HEURISTIC FOR ANALYSIS OF AIRCRAFT SORTIE GENERATION

DEFENSE DATE: 22 February 1994

COMMITTEE:

| Name/Title/Department | Signature |
|---|---|
| *Advisor:*<br>DENNIS C. DIETZ, Lt Col, USAF<br>Assistant Professor of Operations Research<br>Department of Operational Sciences | Dennis C Dietz |
| *Reader:*<br>PETER W. HOVEY<br>Assistant Professor of Statistics<br>Department of Mathematics and Statistics | Peter W. Hovey |



## *Acknowledgments*

I'd like to thank my fellow GOR-94 classmates for making my task of doing this thesis a little easier to bear. Their encouraging words, thoughtful suggestions, and willingness to listen to my complaints got me through the difficult days of thesis despair.

Two individuals, in particular, were of enormous assistance in my work. Alan Wallace, LCOM Simulation Software Engineer for ASC \ ENSSC, patiently explained the mysteries of LCOM to me, answering any and all of my questions. Without Al's help I would have never been able to validate my methodology. Without the help of P. Chandrasekhar Rao, a PhD student at the University of Wisconsin-Madison, I would have never had a methodology to validate. Chandu answered a question I posted to the sci.op-research USENET newsgroup and offered me the results of his PhD research. His work became the basis of my methodology. My debt of gratitude to both is immeasureable.

I can't express how much I appreciate the support I received from my advisor, Lt Col Dennis C. Dietz. He kept me motivated and focused throughout the six months of thesis work. He let me flounder when floundering was okay, he asked me the questions I wasn't smart enough to ask myself, and he rescued me whenever I was drowning.

Finally, I'd like to mention the sacrifice my wife, Myong-Sun, and son, Andy, made during the entire time I attended AFIT. They asked little and gave much. I treasure their love and support.

Richard Carl Jenkins

# *Table of Contents*

## *List of Figures*

## *List of Tables*

## *Abstract*

The primary objective of this study was to develop an analytical methodology based on the Mean Value Analysis algorithm that approximates the performance characteristics of a queueing network model (QNM) containing a fork-join queue with probabilistic branching. These performance characteristics are response time, throughput and queue length at each station. The QNM solved contains the essential features of the aircraft sortie generation process. The sensitivity of the method's accuracy to increases in server utilization was determined. The comparisons of the results of the MVA heuristic to the outputs of the Logistics Composite Model (LCOM) simulation indicate that the heuristic's accuracy decreased as server utilization increases. When server utilization wa kept in realistic ranges, the results of the heuristic for a single fork-join queue were very accurate. For non-maintenance stations, results were within 1 to 2 percent of the LCOM simulation output. For stations on the fork-join queue paths, heuristic results were within 5 percent of the LCOM simulation's output for that portion of the network.

# A MEAN VALUE ANALYSIS HEURISTIC FOR ANALYSIS OF AIRCRAFT SORTIE GENERATION

## *I. Introduction*

### *1.1 Motivation*

Aircraft sortie generation is at the heart of the United States Air Force's mission. The process of aircraft sortie generation includes all those activities necessary for aircraft to fly the maximum number of sorties in a specified period of time. These activities include, but are not limited to, taxiing and flying the aircraft, weapons loading or unloading, and repairing system malfunctions. To successfully accomplish the mission, sufficient resources, such as equipment and personnel, must be available to accomplish each activity in a timely fashion. On the other hand, fiscal contraints limit resource availability. The accurate determination of the resource requirements has historically been accomplished through discrete event simulation of the aircraft sortie generation process. Analytical methods have not been fully explored. Perhaps one reason for this is the difficulty in analytically modeling concurrent maintenance activities.

### *1.2 Background*

The process of aircraft sortie generation can be thought of as the operation of a *system*, a collection of entities (i.e., aircraft, people, equipment and parts) which act and interact together. The system description can be *discrete*, in that the state variables (the collection of variables necessary to describe the system at a

particular time) change only at a countable (or finite) number of points in time (20:2). The aircraft sortie generation process can be completely described by detailing the number of aircraft in various conditions, such as flying, taxiing, or being repaired. The state of the system changes only when an aircraft moves from one condition to another.

*1.2.1 Modeling Techniques.* Because aircraft sortie generation is a discrete system that, as will be discussed below, can be accurately modeled as a queueing network, the most applicable tools for modeling and analyzing the system are discrete-event simulation and queueing theory (16:427) (20:4). Several discrete-event simulations of aircraft sortie generation exist. The Logistics Composite Model (LCOM) is a highly detailed discrete-event simulation that has been validated and "designated" for use by the Air Force (10). For this reason, LCOM is considered to be the standard by which other models of aircraft sortie generation are judged. LCOM and other simulations of aircraft sortie generation are detailed in Chapter 2.

Unfortunately, analysis using a simulation model can be difficult. The exhaustive examination of alternatives using simulation can be very time consuming. This situation is aggravated by the fact a simulation output is random in nature and therefore requires that many iterations be run with identical inputs in order to establish a confidence interval for the results (3:87).

On the other hand, queueing theory can be used to develop queueing network models (QNMs) that do not require a large amount of input data and can execute quickly on a computer (21:80). These models may be analytically solved to obtain such measures of performance as average waiting line length, average waiting time, server utilization, and system throughput. This capability allows the analyst to quickly reveal relationships between key parameters, evaluate tradeoffs and offer fundamental insights into system performance (11:16). For these reasons, QNMs

have been used to analyze several aircraft maintenance-related problems. Some examples of these applications are described in Chapter 2.

The drawbacks of using QNMs are centered around the simplifying assumptions commonly made in order to make the model analytically, i.e., mathematically, tractable. Simplifying assumptions are made concerning the customer arrival and service time distributions, queueing discipline, scheduling policies, and steady-state conditions (8:25). The structure of the model itself may prevent the use of analytical methods for arriving at exact results, but useful approximate solutions can often be obtained.

### *1.3 Problem*

Aircraft sortie generation is fairly independent of the aircraft type with differences confined largely to the maintenance aspects of the process. Thus, while the number of aircraft systems and their probability of failure is unique for each aircraft type, the overall process of sortie generation remains basically invariant.

The process begins with the aircraft taxiing to the runway for takeoff. The pilot conducts system checks during ground operation and, with a small probability, some system or systems may fail and require maintenance, thereby aborting the sortie. If the aircraft flies, it will do so for a predetermined length of time. With a higher probability than that of at the end-of-runway check, some aircraft system or systems may have failed in flight and require maintenance upon landing. If no maintenance is required upon landing, the aircraft is "turned" (i.e., prepared for its next sortie) rearmed with weapons, and again taxis to the runway for its next sortie. If maintenance is required due to system failures at the end of the runway, the aircraft's weapons must be dearmed or removed before the maintenance can begin. Maintenance, regardless of when the failure occurred, is performed on all malfunctioning systems simultaneously. Each system has a unique probability of

failure. Upon completion of all maintenance, the aircraft is turned, rearmed, and taxis for its next sortie.

The entire process is illustrated in figure 1.1. Each activity of the sortie generation process can be modeled as a station in a queueing network. Each station is numbered, with $P_{ij}$ representing the probability that the aircraft transitions from queue $i$ to $j$. The service times, $s_i$, and number of servers, $n_i$, for each queue are listed in Chapter 4 where the network is solved. These network parameters are notional; only the QNM structure reflects an actual aircraft sortie generation process.

Five aspects of the aircraft sortie generation process are difficult to model and solve using queueing theory. These are:

1. Concurrent Maintenance: Upon experiencing a ground abort or a sortie completion, an aircraft may have more than one malfunctioning system. These systems could all be repaired simultaneously by personnel with separate maintenance specialties. This aspect of the system is modeled with a fork-join queue. The fork-join queue, illustrated in figure 1.2, has multiple paths emanating from a fork node with each path representing a separate maintenance speciality. The maintenance activities performed by the speciality are modeled as a multiple-server queue. The number of servers represents the number of personnel available in that speciality and the service time is the average completion time of the maintenance activities performed by the speciality. The probabilities, $P_j^{fj}$, are the conditional probabilities that maintenance speciality $j$ must perform a maintenance activity given that the aircraft has some malfunctioning system.
2. Maintenance Crew Sizes: Personnel from a given maintenance specialty may be responsible for more than one aircraft system. Repair of these systems may require different service times and crew sizes. This aspect of maintenance is modeled within the fork-join queue by averaging crew sizes and completion

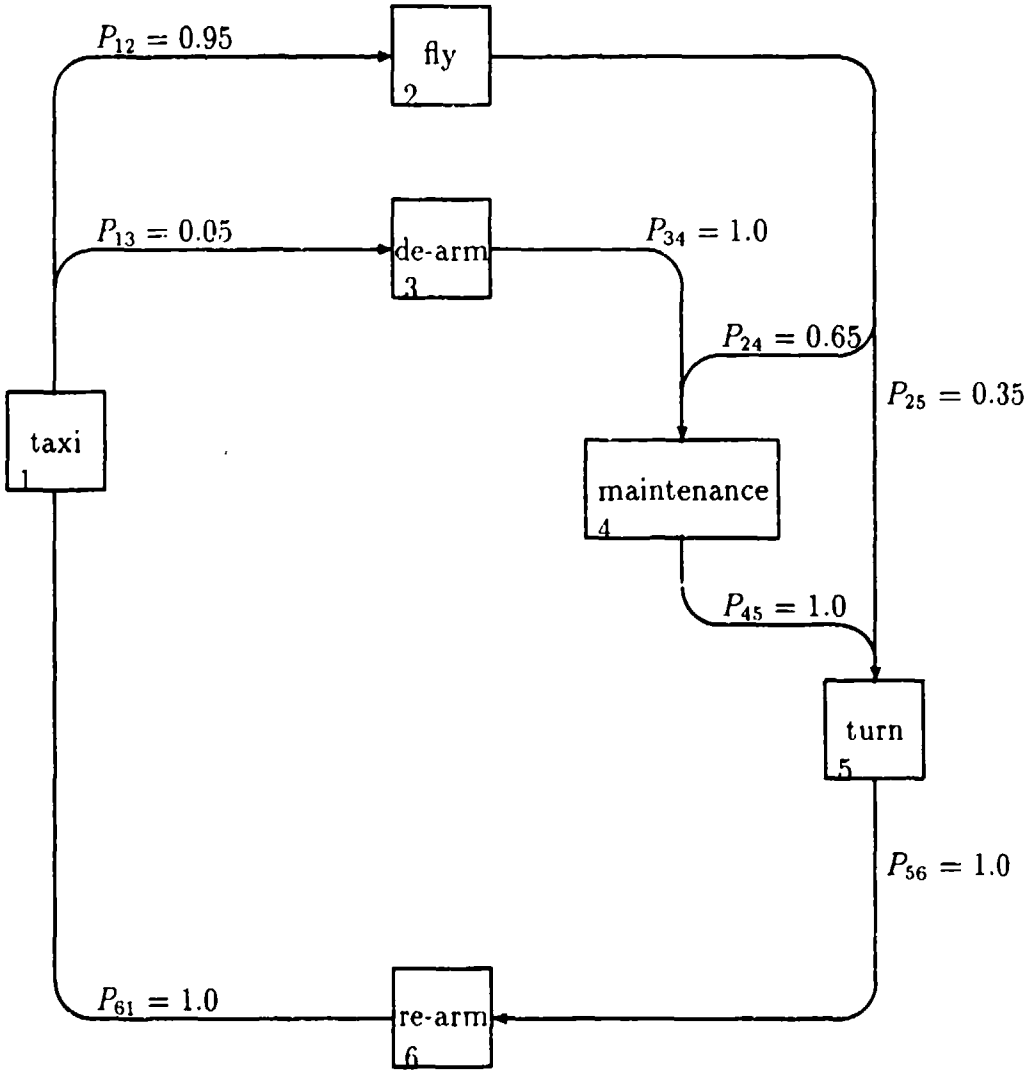



Figure 1.1. Queuing Network Model of Aircraft Sortie Generation

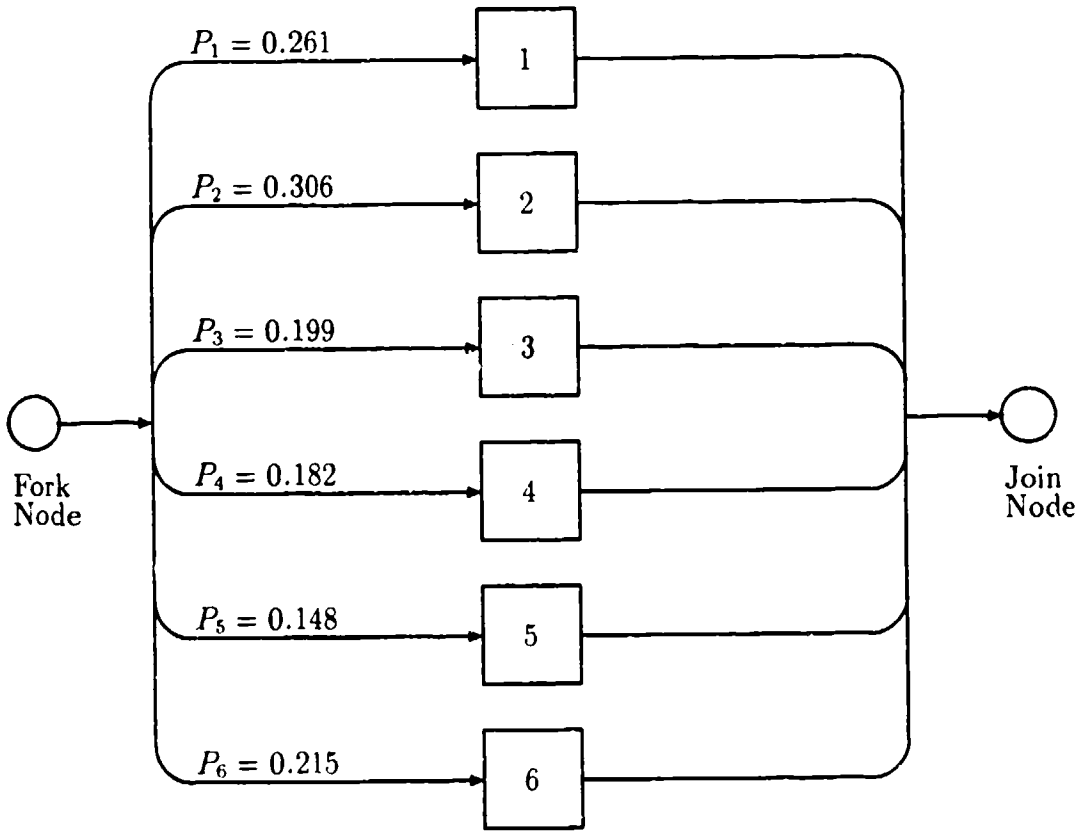



Figure 1.2. Fork-Join Queue for Maintenance Activities

times over all the activities that each maintenance speciality must perform. For example, if a maintenance speciality has 20 people available and the average crew size is 2, then the multiple-server queue modeling that speciality has 10 servers and each server is considered two people. An alternative method of addressing this issue is known as the machine-operator interference problem and is discussed next.

3. Machine-Operator Interference: Some maintenance tasks, such as de-arming aircraft weapons before ground-abort maintenance and rearming an aircraft prior to flight, occur at different points in the network but utilize the same maintenance manpower specialty. This linkage of resources is referred to in the literature of queueing network theory as machine-operator interference. In this study, the issue is handled by treating the different tasks as separate queues with their own resource pool. However, machine-operator interference in automatic assembly systems with some similarities to the aircraft sortie generation process has been sucessfully modeled (7:275) (17:93).

4. Batch Arrivals: Some combat aircraft rarely fly alone, but rather in groups (flights) of two or four. The simultaneous arrival of multiple customers to a fork-join queue is an aspect of the theory that has yet to be addressed in the literature and is, therefore, not modeled in this study.

5. Mission Scheduling: Aircraft normally do not fly immediately after maintenance is complete, but instead fly according to a predetermined schedule. The time aircraft spend waiting for a scheduled take-off time reduces the flow of aircraft through the network. Having aircraft fly sorties as soon as they can creates a more demanding sortie generation scenario. Maintenance personnel that can support a "fly when ready" scenario can easily support a scenario with delays due to scheduling. Thus, mission scheduling is not modeled in this study.

## *1.4 Objective*

The primary objective of this study is to develop an analytical methodology based on the Mean Value Analysis algorithm that approximates the performance characteristics of a queueing network model containing a fork-join queue with probabilistic branching. These performance characteristics are response time, throughput and queue length at each queue. The QNM to be solved contains the essential features of the aircraft sortie generation process. The sensitivity of the method's accuracy to increases in server utilization will be determined.

*1.4.1 Approach.* The objectives of the study are accomplished through the following steps.

1. Develop a solution method based on the Mean Value Analysis algorithm that approximates the solution of an analytical queueing network model containing a fork-join queue with probabilistic branching.
2. Solve the QNM in figure 1.1 with the fork-node queue in figure 1.2 representing maintenance.
3. Develop an LCOM-version of the same network. This network will be simulated in LCOM with a sufficient number of iterations to guarantee a reasonably tight confidence interval for the simulation's outputs.
4. Compare the analytical solution of the QNM to the LCOM simulation output.
5. Repeat steps 2 thru 4 for the QNM with specific network parameters changed such that server utilizations are increased.

*1.4.2 Scope.* Aircraft maintenance is divided into two categories depending upon where the maintenance is performed. These two categories are *on-equipment* and *off-equipment maintenance*, where the term *equipment* refers to the aircraft. On-equipment maintenance entails those tasks where the malfunctioning aircraft system is repaired without leaving the aircraft. Examples include avionics systems

troubleshooting, component removal and replacement and airframe repair. Off-equipment maintenance is performed on systems that have been removed from the aircraft and taken to a workcenter for repair. The most common instance of off-equipment maintenance is the repair of an avionics system in the corresponding workcenter. This study will address on-equipment aircraft maintenance only.

# *II. Previous Work*

## *2.1 Introduction*

Previous work in the modeling of aircraft sortie generation, regardless of the purpose of the modeling, has primarily employed simulation methods. However, some examples of analytical queueing models also exist.

## *2.2 Simulation Models*

Most military studies of aircraft sortie generation have relied on simulation as the principal evaluation tool (24). This is primarily due to the complexity of the maintenance systems that comprise a large part of the process (11:2). For example, the repair of a single malfunctioning aircraft system may require the combined efforts of several different maintenance specialities working in sequence or even simultaneously. Each of these maintenance specialities has a different size manpower pool from which to draw, a unique set of aircraft systems whose repair are its responsibility, and its own constantly changing backlog of aircraft malfunctions waiting to be repaired. Simulation offers the unique ability to capture the complexities of systems that cannot be otherwise analyzed. Some recent simulations of the aircraft sortie generation process are discussed below.

*2.2.1 The Logistics Composite Model.* The Logistics Composite Model (LCOM) is the aircraft sortie generation model that has been approved for use in the Air Force. It is a discrete event simulation written in Simscript II.5. LCOM's original purpose, when addressed by the Rand Corporation in the late 1960s, was to provide an analysis tool to relate base-level logistics resources (the model's inputs) with each other and with sortie generation rates (the model's output) (14:2). The Air Force now uses the model primarily to determine maintenance manpower requirements (12:1-1) (27:1) (5:3).

Maintenance resource levels, i.e., spare parts, people and equipment, are established by the analyst. To obtain the minimum mix of resources required to sustain the desired sortie rate, resources are constrained until the sortie rate is affected (5:10). However, because LCOM's results inherently contain the randomness of the Monte Carlo simulation process, the optimal solution is inexact.

LCOM's size and complexity limit its usefulness. Data input is difficult and time consuming; for a detailed weapon system study, an LCOM data base can run to several thousand lines of code (14:75) (27:2). LCOM runs are also time consuming; a single run from a large database can take over three hours. Since LCOM is run iteratively, with the analyst varying inputs in order to determine changes in output, LCOM studies can be several months in duration.

LCOM continues to be updated and refined in response to the demands of long term use. The model has been made increasingly user- friendly and, if run on modern computers, is relatively fast running (5:14). LCOM is used by the United States Air Force's Air Combat Command, Air Mobility Command and Air Material Command (29).

*2.2.2 LMI Sortie Generation Model.* In the late 1970s, The Logistics Management Institute (LMI), working for the Assistant Secretary of Defense for Manpower, Reserve Affairs, and Logistics, developed the Sortie Generation Model (SGM). The model was developed to estimate the effect of varying levels of certain classes of resources on the readiness of tactical fighter forces. The SGM system consists of the model itself and a complex system of supporting software that produces information files necessary to run the SGM such as spares availability and the maintenance environment (1:6-7).

Scenario characteristics are specified interactively by the user. Characteristics such as the number of waves or flying periods per day, the attrition rate, the number of aircraft in reserve, and the ground abort rate can be set to vary daily. Other

characteristics such as the sortie length, the probability that a returning aircraft requires maintenance and the first and last takeoff times of the day remain constant throughout the scenario. The SGM, a hybrid analytic/simulation, then estimates maximal sortie-generation capability across time as a function of its input files and scenario characteristics (1:8).

SGM accomplishes this estimation by tracking the number of aircraft in each of five states: 1) Mission-capable; 2) Maintenance; 3) Not mission-capable, supply; 4) Combat loss; 5) Reserve. There are eight processes by which aircraft move from state to state. These are determined by the input files and scenario characteristics and include ground aborts, breakage, repair, attrition, and parts demands. Aircraft availability is assessed against the daily flying schedule (2:1-3).

Aircraft system malfunctions in the SGM are determined by a series of random draws from binomial distributions. An initial draw is made to first determine if the aircraft has any breakage. Additional draws are made to determine what maintenance workcenters have malfunctioning equipment that require repair (2:2-3). The malfunctioning equipment is not specified. Maintenance is modeled as a multiple-server queuing process with aircraft tracking accomplished via attribute marking (2:2-7).

*2.2.3 TSAR.* TSAR is a complex Monte Carlo simulation, written in FORTRAN, of a system of interdependent theater airbases. TSAR was developed by the Rand Corporation in the early 1980s for analyzing the interrelations among on-base resources and the capability of the airbases to generate aircraft sorties in a wartime environment. On-equipment maintenance tasks, parts and equipment repair jobs, munitions assembly and facility repair tasks are simulated for each of several airbases. Asset accounting for each of 11 classes of resources, with hundreds of resources in each class, permits assessment of a broad range of policy options that could improve the efficiency of resource utilization on a theater-wide basis (13:v).

TSAR can also model the effects of conventional or chemical airbase attacks, the effects of having personnel operate in individual chemical protection equipment, and the efforts to reconstitute the airbase following an attack (23:4).

TSAR is a large and complex model requiring a lot of upkeep. There are no data preprocessors for TSAR that are able to take pre-collected data and transform it into TSAR format; thus, the process of data preparation and input can be very time consuming (23:6). Just as in the case of LCOM, TSAR's results inherently contain the randomness of the Monte Carlo simulation process; several iterations of the model must be run in order to establish a confidence interval for the output data.

*2.2.4 Dyna-Sim.* Dyna-Sim is a discrete-event simulation developed by the Rand Corporation in the mid-1980s. The model, written in Simscript II.5, was built to simulate the queueing of aircraft maintenance jobs for automatic test equipment (ATE) in order to quantify the importance of ATE in the repair cycle (22:iii). The simulated repair system is a multi-server, multi-job-class queue with time-varying arrival rates. The simulated queue is multi-server because more than one ATE is normally available for repair actions. The multi-job-class aspect of the model captures the fact that the ATE is used to repair more than one type of malfunctioning system. Jobs are selected from the maintenance queue using four different priority rules (22:6).

While the scope of the Dyna-Sim model is limited to only the ATE issue, it could be easily adapted to model any maintenance repair activity with multiple servers and multiple job-classes. However, the analyst must specify the various job arrival rates, i.e., the system does not model the closed network aspects of aircraft sortie generation. The analyst specifies when arrival rates change rather than have these changes dependent upon repair frequencies, service times, and queue lengths (22:3).

## 2.3 *Analytical Models*

Analytical models have been little used in the modeling of aircraft sortie generation. Their use has been primarily confined to small aspects of the larger system such as specific manpower or equipment issues. Some examples of analytical methods applied to the aircraft sortie generation process are discussed below.

*2.3.1 Fleet Maintenance.* Boling and Hillier used queueing theory to analyze fleet maintenance systems, where the fleet consists of trucks, aircraft or cars on a scheduled maintenance program as well as emergency repair basis (4:1). They developed a general queueing network model for the design of fleet maintenance systems involving several crews working in sequence. Emphasis was placed on optimizing the system for scheduled periodic maintenance activities. The optimal system configuration was determined as a function of the number of crews, cost of crew activities and cost of a fleet unit's idle time (4:2). In addition to the assumption of sequential tasks, the method presented is also based on the assumption that the expected service times for the sequential tasks are equal. These assumptions could generally not be made in developing a model of a sortie generation process for military aircraft.

*2.3.2 B-52H/KC-135 Maintenance Model.* A 1989 study, conducted by the Air Force Logistics Command, examined the maintenance manpower, spares cost and repair cost impacts on going from three to two levels of maintenance on avionics spares for the B-52H Bomber and the KC-135 Tanker. Queueing theory was used to show that consolidating several base maintenance shops into a single repair facility would reduce the maintenance manpower needed to give the same or better service (25:1). This was achieved by bounding the number of servers required in the repair facility from above and below. The upper bound was determined by solving an M/M/s queue for the minimum number of servers required to ensure that the expected waiting time under consolidation was less than or equal to the base expected waiting time. The lower bound was determined by combining servers into

a consolidated server with a service rate equal to the base service rate multiplied by the number of servers combined. The expected waiting time was held constant and the resulting M/M/1 queue was solved for the fraction of servers that must be retained under consolidation (25:45-49). While a heuristic argument for the acceptance of these bounds was presented, no formal proof or validation of the bounds was provided.

*2.3.3 Optimal Specialization.* Dietz and Rosenshine developed an analytical method for determining an optimal specialization strategy for a maintenance manpower force. The model assumed that maintenance tasks were generated by a system of identical machines, such as aircraft, which experience random malfunctions and require periodic service. As a part of the overall method, the impact of alternative manpower structures on system performance was evaluated using a queueing network model (11:2). The method requires the simultaneous solution of the network's global balance equations, thus limiting the applicability of the approach to simple systems or highly aggregated analysis of complex systems (11:16).

*2.3.4 SUMMA.* The Small Unit Maintenance Manpower Analyses (SUMMA) method was developed in the late 1980s for the Air Force Human Resource Lab. The method was originally intended to be used for gauging the impacts of airbase dispersal in wartime on manpower requirements and unit performance under proposed alternative arrangements of maintenance job/task skills (6:i). The SUMMA method grew into a comprehensive system that would consider maintenance, manpower, personnel and training (MPT), and cost factors to help the Air Force improve it maintenance task allocation/specialty structure. The system uses maintenance task and MPT data for specific operational and maintenance scenarios in a decision support system (DSS) to expose the tradeoffs and implications of occupational specialty consolidation and/or other maintenance task/specialty changes (30:6). The SUMMA DSS analyzes the maintenance process in terms of task completion times. Using

the method of Lagrangian Multipliers, the number of personnel in each speciality is minimized subject to maximum aircraft downtime constraints. Thus a task/specialty assignment that achieves the desired sortie rate using the minimum manpower while meeting these user-specified constraints is derived (30:99-110). The operational capability of the task allocation is then verified using LCOM.

The SUMMA method contains a major assumption that does not reflect real-world practice. In the SUMMA task allocation model it is assumed that all tasks on an aircraft are performed sequentially (30:104). This assumption is enforced even if the aircraft requires maintenance on separate systems maintained by separate specialities that could work on the aircraft simultaneously. In actual practice, these systems would be repaired at the same time. The SUMMA model itself relies on LCOM to evaluate and verify the task allocation derived by the SUMMA method. The SUMMA method produces "...task/AFS allocations...judged to have flaws that cannot be accepted", thus the method "...indeed expects the user to revise and refine his solution a number of times" (30:21).

# III. Methodology

## 3.1 Introduction

The method used in this study to analyze the aircraft sortie generation process is a heuristic based upon the Mean Value Analysis (MVA) algorithm. The MVA algorithm, which yields exact results for closed networks that have a product form solution, can be modified to produce approximate results for networks which do not have product form solutions, such as a sortie generation model (18:33). A discussion of the accuracy of the heuristic is postponed until Chapter 4 when output of the LCOM simulation of the network is compared to the results attained from the MVA heuristic. The terms "queue" and "station" are used interchangably throughout this chapter.

## 3.2 The MVA Algorithm

The MVA algorithm for closed networks with product form solutions is a recent development in queueing theory, first published in 1980 (19). It allows us to analyze the steady-state behavior of a closed network that exhibits flow balance and one-step behavior. A network exhibits *flow balance* if the total number of arrivals to a station during a period $T$ equals the total number of departures from the station. *One-step behavior* means that arrivals do not coincide with departures and at any instant only one arrival or one departure can occur (18:89).

A closed network consists of $N$ customers distributed at $K$ interconnected queues (stations). The $K$ queues may be single-server (load-independent) or multiple-server (load-dependent). The service rate at a station $i$ is denoted $\mu_i$, while the service time, the inverse of the service rate, is denoted $s_i$. $P_{ij}$ is the probability that a customer completing service at station $i$ will then proceed to station $j$. The response time of a station $i$, $R_i$, measures the combined waiting and service time for a customer. The queue length, $Q_i$, is the number of cutomers waiting for service and

being served at station $i$. The utilization of station $i$, $U_i$, has two interpretations. If station $i$ is a single-server queue, it is the average percentage of time the server is busy or, if the station is a multiple-server queue, it is the average number of busy servers.

The MVA algorithm yields the mean values of the network performance characteristics of response time, queue length, throughput, and utilization for each station in the network. Mean values are normally the most desired outputs of a model (15:122). Simply put, the algorithm determines the mean response time of a station $i$ in a network to be the sum of the mean service times and the mean waiting times a customer experiences before being served. This relationship is based on two fundamental results: the Arrival Theorem and the Marginal Local Balance Theorem.

The Arrival Theorem tells us that an arriving customer sees a random observer's distribution with the arriving customer removed from the network. This can be stated as:

$$PA_i(n|N) = P_i(n|N-1) \tag{3.1}$$

where $PA_i(n|N)$ is the probability that, in a closed network containing $N$ customers, an arriving customer to station $i$ finds $n$ customers already there and $P_i(n|N-1)$ is the probability that a random observer sees $n$ customers at station $i$ when there are $N$ - 1 customers in the network (18:97).

From the Arrival Theorem we get the Marginal Local Balance Theorem, which can be interpreted as a detailed balance between two adjacent states for station $i$ as shown in figure 3.1.

The rate of departure from state $(n|N)$ is $\mu_i(n)$, whereas the rate of entry into it from state $(n-1|N-1)$ is the throughput $\lambda_i(N)$, yielding a local balance equation of

$$\mu_i(n)P_i(n|N) = \lambda_i(N)P_i(n-1|N-1)$$

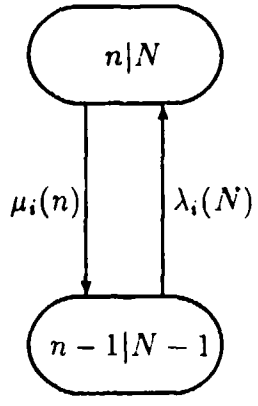



Figure 3.1. Marginal Local Balance for Station $i$

We also note that

$$\mu_i(n) = \mu_i^o C_i(n) = \frac{C_i(n)}{s_i^o}$$

where $\mu_i^o$ is the basic service rate and $s_i^o$ is the basic service time for a server at station $i$ and $C_i(n)$ is the number of servers busy when $n$ customers are at station $i$ (18:232). From this we get a statement of the Marginal Local Balance Theorem:

$$P_i(n|N) = \frac{s_i^o \lambda_i(N)}{C_i(n)} P_i \quad -1|N-1) = \frac{U_i(N)}{C_i(n)} P_i(n-1|N-1) \tag{3.2}$$

where the last equation makes use of the Utilization Law (18:94,102),

$$U_i(n) = s_i^o \lambda_i(n)$$

This theorem can be used to derive several other relationships that are useful in computing performance measures recursively. For example,

$$Q_i(N) = \sum_{n=1}^{N} n P_i(n|N) = U_i(N) \sum_{n=1}^{N} \frac{n}{C_i(n)} P_i(n-1|N-1) \tag{3.3}$$

Recalling Little's Law for a station $i$ when $n$ customers are present,

$$Q_i(n) = \lambda_i(n) R_i(n)$$

and again making use of the Utilization Law (18:232), we substitute Equation 3.3 into Little's Law and solve for the response time to yield

$$R_i(N) = s_i^o \sum_{n=1}^{N} \frac{n}{C_i(n)} P_i(n-1|N-1) \tag{3.4}$$

If station $i$ has only one server then $C_i(n) = 1$ for all $n$ and the same substitution yields,

$$R_i(N) = s_i \sum_{n=1}^{N} n P_i(n-1|N-1) = s_i[1 + Q_i(N-1)] \tag{3.5}$$

where $Q_i(0) = 0$. Note that if station $i$ has more servers than potential customers, the station is referred to as a *delay station.* In this situation, Equation 3.5 yields $R_i(n) = s_i$.

Equation 3.5 is known as the *mean value theorem.* Equations 3.4 and 3.5 are at the heart of the MVA Algorithm. By relating the response time of a station when $n$ customers are present to the length of the queue at the station when $n$ - 1 are present, we are able to iteratively determine each station's steady state queue length, response time, and utilization. In order to make these determinations, it is necessary to calculate the average cycle time for a customer when $n$ customers are in the system. A customer at station 1 is chosen as reference. The time between two departures by the same customer from station 1 is:

$$T_1(n) = \sum_{i=1}^{M} \frac{v_i R_i(n)}{v_1} \tag{3.6}$$

where $\frac{v_i}{v_1}$ is the mean number of visits customers make to station $i$ for every visit to station 1. The values $v_i$ are referred to as *visit ratios.* Visit ratios are determined

by solving the system of equations $\mathbf{v} = \mathbf{v}P$ where $P$ is the matrix of transition probabilities for the network. In order to solve this system, one of the $v_i$ can be set to an arbitrary value (normally $v_1 = 1$).

The throughput for station 1, when $n$ customers are in the network, is

$$\lambda_1(n) = \frac{n}{T_1(n)} \tag{3.7}$$

The ratio $\frac{v_i}{v_1}$ determines the amount of station 1 throughput at station $i$. Thus, $\lambda_1(n)$ is used to determine the queue length and utilization at each station as follows:

$$Q_i(n) = R_i(n)\lambda_1(n)\frac{v_i}{v_1} \tag{3.8}$$

and

$$U_i(n) = s_i\lambda_1(n)\frac{v_i}{v_1} \tag{3.9}$$

If station $i$ has a single server, then the result of Equation 3.8 can now be applied to Equation 3.5 to begin the next iteration of the MVA algorithm. If station $i$ has multiple servers, the result of Equation 3.9 and the Marginal Local Balance Theorem can be used to find $P_i(n-1|N-1)$ as follows:

$$P_i(k|n) = \frac{U_i(n)P_i(k-1|n-1)}{C_i(k)} \tag{3.10}$$

and

$$P_i(0|n) = 1 - \sum_{k=1}^{n} P_i(k|n) \tag{3.11}$$

These two results can now be used in Equation 3.4 to begin the next iteration of the MVA Algorithm.

In summary, the MVA algorithm begins by noting that $Q_i(0) = 0$ so that $R_i(1) = s_i$. Equations 3.6 thru 3.9 are calcualted to determine the values of the mean performance characteristics of each station. Equations 3.10 and 3.11 are calculated

for multiple server stations. The next iteration of the MVA algorithm, i.e., when the number of customers in the networks is $n = 2$, uses the information of the previous iteration as detailed above. Iterations continue, updating the values of the mean performance characteristics at each station with each iteration, until $n = N$. The MVA algorithm is then complete and the final values for each station's response time, throughput, queue length, and utilization represent the mean performance characteristics of the network under study.

### *3.3 An MVA Heuristic for Fork-Join Queues*

In order to model maintenance activities occurring simultaneously within the aircraft sortie generation process, it is necessary to add a fork-join queue to the closed queueing network. The new network no longer possesses a product form solution, so the MVA algorithm can not be applied (19:1048). Recently, however, a heuristic based on the MVA was developed by Rao and Suri to deal with this case (26). The following discussion summarizes their work.

The fork-join queue in the network studied by Rao and Suri consists of an independent single server queue on each path emanating from a fork node. Service times of the queues need not be identical. Each customer in the network takes each path from the fork node. A customer cannot leave the join node until the service activities of each path are completed for that customer. That is, a parent customer breaks into statistically identical sibling customers at the fork node, one sibling per path. Each sibling customer finishing service must wait until all other sibling customers have completed their service on their fork path. Once the sibling-customers have all completed service, they are re-joined into the parent customer at the join node. A typical fork-join queue is illustrated in figure 3.2 below. Stations 1 to $K$ represent the $K$ single-server queues on the paths of the fork-join node while station $K + 1$ is a single-server queue that comprises the remainder of the network.

The $H_i$ represent the queueing areas and the $D_i$ are the waiting areas after service for the $K$ paths.

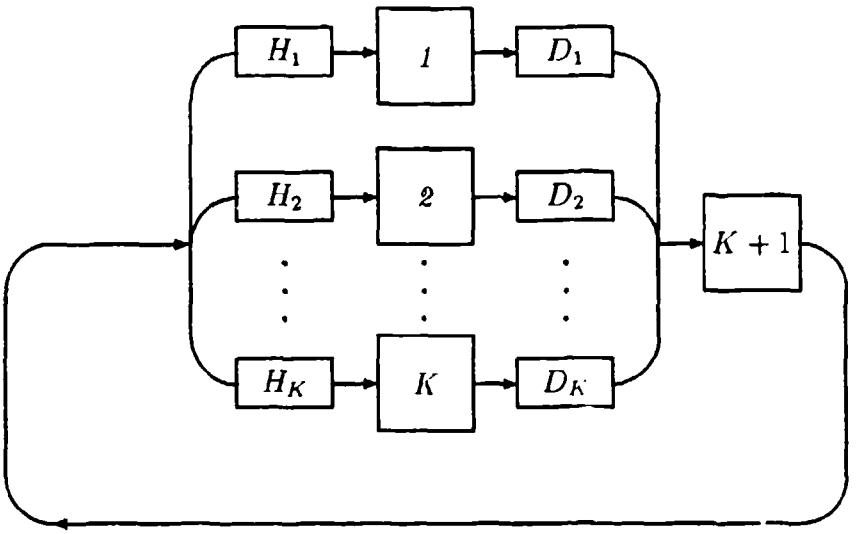



Figure 3.2. Closed Network with Fork-Join Queue

The MVA heuristic for fork-join queues makes two key assumptions (26:8). These two assumptions are exact when only one customer is in the network. When more than one customer is present these assumptions become approximations, hence the MVA algorithm becomes a heuristic. The first assumption is a heuristic application of the Arrival Theorem. It supplies the pivotal equation used in the iterative calculations of the mean peformance characteristics of the network.

(A1) An arriving customer to a path of the fork-join queue sees a random observer's distribution with itself removed from the network (that is, removed from each of the fork-join paths).

This can be expressed as:

$$PA_i(\mathbf{n}) = P_i(\mathbf{n} - \mathbf{1}) \tag{3.12}$$

where $\mathbf{n}$ is the population vector for the fork-join queue, with $n_k = n \ \forall k$ and $\mathbf{n} - \mathbf{1}$ is the population vector for the fork-join queue with $n_k = n - 1 \ \forall k$. As with the MVA algorithm this leads to

$$R_i(\mathbf{n}) = s_i[1 + Q(\mathbf{n} - \mathbf{1})] \tag{3.13}$$

where $Q_i(\mathbf{0}) = 0$. Because Rao and Suri considered only the case where all stations, station $K + 1$ as well as the fork-join paths, are single-server queues, a load-dependent version of Equation 3.13 is not needed (26:1). Their heuristic will be expanded to cover multiple-server queues on the fork-join paths in the next section.

The second assumption is critical to determing the length of time sibling customers must wait until they can be re-joined into the parent customer at the join node.

(A2) The response time distribution of a customer at station $k$ is exponential and independent of the response time of its sibling customers on the other paths.

Using fork-join queue path 1 as reference, we can determine how long sibling customer 1 has to wait until all of its sibling customers complete service and then may be rejoined into the parent customer. The probability that sibling customer 1 must wait for sibling customer 2 is merely the probability that sibling customer 1 finishes service before sibling customer 2. Using the well known formula for calculating this probability when the service rates are exponential, we have

$$p_2(\mathbf{n}) = \frac{m_1(\mathbf{n})}{m_1(\mathbf{n}) + m_2(\mathbf{n})} \tag{3.14}$$

where $m_i$ is the service rate of fork-join queue path $i$. Assumption (A2) allows us to apply the memoryless property of the exponential distribution to determine that the *time* sibling customer 1 must wait for sibling customer 2 is $R_2(\mathbf{n})$.

Continuing to use fork-join queue path 1 as reference, the probability that sibling customers 1 and 2 must wait for sibling customer 3 is

$$p_3(\mathbf{n}) = P\{t_3(\mathbf{n}) > \max[t_1(\mathbf{n}), t_2(\mathbf{n})]\}$$

where the $t_i(\mathbf{n})$ are independent exponential random variables with rates $m_i(\mathbf{n})$. The cumulative distribution function of $Y_2 = \max[t_1(\mathbf{n}), t_2(\mathbf{n})]$ is given by:

$$F_{Y_2}(x) = (1 - e^{-m_1(\mathbf{n})x})(1 - e^{-m_2(\mathbf{n})x})$$

Differentiating with respect to $x$, we obtain the probability density function of $Y_2$.

$$f_{Y_2} = m_1(\mathbf{n})e^{-m_1(\mathbf{n})x} + m_2(\mathbf{n})e^{-m_2(\mathbf{n})x} - [m_1(\mathbf{n}) + m_2(\mathbf{n})]e^{-[m_1(\mathbf{n})+m_2(\mathbf{n})]x}$$

Thus,

$$p_3(\mathbf{n}) = P\{t_3(\mathbf{n}) > Y_2\} = \int_0^\infty P\{t_3(\mathbf{n}) > x\} f_{Y_2}(x)\, dx$$

Now, since $t_3(\mathbf{n})$ is exponentially distributed with rate $m_3(\mathbf{n})$, we have

$$\begin{aligned} p_3(\mathbf{n}) \;=\; & \int_0^\infty \{e^{-m_3(\mathbf{n})}[m_1(\mathbf{n})e^{-m_1(\mathbf{n})x} + m_2(\mathbf{n})e^{-m_2(\mathbf{n})x} \\ & -[m_1(\mathbf{n}) + m_2(\mathbf{n})]e^{-[m_1(\mathbf{n})+m_2(\mathbf{n})]x}]\}\, dx \end{aligned}$$

so that

$$p_3(\mathbf{n}) = \frac{m_1(\mathbf{n})}{m_1(\mathbf{n}) + m_3(\mathbf{n})} + \frac{m_2(\mathbf{n})}{m_2(\mathbf{n}) + m_3(\mathbf{n})} - \frac{m_1(\mathbf{n}) + m_2(\mathbf{n})}{m_1(\mathbf{n}) + m_2(\mathbf{n}) + m_3(\mathbf{n})} \qquad (3.15)$$

Again using Assumption (A2), we determine the waiting time to be $R_3(\mathbf{n})$. For each of the $k$ paths, the probabilities $p_i(n)$ and average waiting times $R_i(\mathbf{n})$ that sibling customers 1 through $i$ - 1 must wait for sibling customer $i$, with $i = 2, 3, \ldots, k$. are calculated in a similar manner (26:10-12).

We are now able to calculate the mean time a customer spends in the system. Considering the subnetwork consisting only of fork-join path 1 and station $K+1$, the time between two departures from station 1 for the same customer when $\mathbf{n}$ customers are in the subnetwork is

$$T_1(\mathbf{n}) = R_1(\mathbf{n}) + \sum_{i=2}^{K} p_i(\mathbf{n}) R_i(\mathbf{n}) + R_{K+1}(\mathbf{n}) \tag{3.16}$$

We can now apply Little's law to this subnetwork to get

$$\lambda_1(\mathbf{n}) = \frac{n}{T_1(\mathbf{n})} \tag{3.17}$$

Since arrivals to the fork-join queue paths are simultaneous, the throughputs of all the subnetworks consisting of a single fork-join path and station $K+1$ are identical. That is,

$$\lambda_1(\mathbf{n}) = \lambda_2(\mathbf{n}) = \cdots = \lambda_K(\mathbf{n}) = \lambda(\mathbf{n}) \tag{3.18}$$

where $\lambda(\mathbf{n})$ is the network throughput when $\mathbf{n}$ customers are present. We also note from the last two equations that

$$T_1(\mathbf{n}) = T_2(\mathbf{n}) = \cdots = T_K(\mathbf{n}) = T(\mathbf{n}) \tag{3.19}$$

which justifies our use of the subnetwork consisting only of fork-join path 1 and station $K+1$ as reference.

Each station's utilization and queue length can be calculated as

$$U_i(\mathbf{n}) = s_i \lambda(\mathbf{n}) \tag{3.20}$$

and

$$Q_i(\mathbf{n}) = R_i(\mathbf{n}) \lambda(\mathbf{n}) \tag{3.21}$$

The resutlts of Equation 3.21 can now be substituted into Equation 3.13 to begin the next iteration of the MVA heuristic for fork-join queues.

Just as in the MVA algorithm, the MVA heuristic for fork-join queues begins by noting that $Q_i(\mathbf{0}) = 0$ so that $R_i(\mathbf{1}) = s_i$. Next, $T_1(\mathbf{1})$ is calculated, allowing the throughput when one customer is in the system to be determined. The utilization and queue length of each station is then calculated using Equations 3.20 and 3.21. This information is used in Equation 3.13 in the next iteration for $\mathbf{n} = \mathbf{2}$ customers. Iterations continue until $\mathbf{n} = \mathbf{N}$ when the mean performance characteristics of each station will have been determined.

### *3.4 Extension to Fork-Join Queues with Probabilistic Branching*

A QNM of aircraft sortie generation is much more complicated than the basic fork-join QNM discussed above. The method used to solve the aircraft sortie generation QNM, while similar to that of the basic fork-join QNM, must account for these complications. For example, in the above calculations, we assumed that a customer took each path of the fork-join queue. In an aircraft sortie generation QNM, where the fork-join queue represents the maintenance activities that can occur on the aircraft simultaneously, branching at the fork node is probabilistic. That is, when an aircraft has a malfunction, it may have any combination of system malfunctions. Thus, the aircraft "customer" may fork into sibling customers that take fork-join paths 1, 2, and 4 or paths 3 and 4 or paths 1, 2, 5 and 6 and so forth. We must account for this probabilistic branching in determining the queue lengths and server utilizations for each fork-join path.

The response time of queueing stations outside of the fork-join queue are determined using the MVA algorithm equations for single-server or multiple-server queues as appropriate. The response time of each fork-join queue path is calculated using

MVA algorithm Equation 3.4 for load-dependent stations,

$$R_i(\mathbf{n}) = s_i^o \sum_{n=1}^{N} \frac{n}{C_i(n)} P_i(n-1|N-1)$$

and not Equation 3.13 for load-independent stations, as in the network studied by Rao and Suri. This is because the basic fork-join QNM and the aircraft sortie generation QNM differ in the number of servers on each path of the fork-join queue. Each aircraft maintenance speciality will have a pool of personnel to draw from. Thus, instead of a single-server queue on each path, the aircraft sortie generation fork-join queue has a multiple-server queue on each path.

Due to the probabilistic branching of the fork-join queue, the cycle time of a representative customer cannot be determined as in the MVA heurisitic developed by Rao and Suri. Recall that they calculated the cycle time from a single subnetwork consisting of a representative fork-join queue path and the rest of the network outside the fork-join queue. We must determine this cycle time for $K$ different subnetworks, one for each fork-join queue path and the rest of the network outside the fork-join queue.

We begin by deterimining the average response time of the fork-join queue given that a customer takes path $i$. From the first two terms of Equation 3.16 we can determine the time that a sibling customer spends on fork-join queue path $i$ waiting to be served, being served, and waiting for its other sibling customers to finish their service. This time is:

$$T_i(\mathbf{n}) = R_i(\mathbf{n}) + \sum_{j \neq i} p_j(\mathbf{n}) R_j(\mathbf{n}) P_j^{fj} \tag{3.22}$$

where $j = 1, 2, \ldots, K$. As discussed in the previous section, $p_j(\mathbf{n})$ is the probability that sibling customers 1 through $j$ - 1 must wait for sibling customer $j$ and $R_j(\mathbf{n})$, the response time on fork-path $j$, is how long they must wait. The term $P_j^{fj}$ is the

probability that fork-join path $j$ is taken, given that at least one of the paths are taken.

The cycle time for station 1 of a representative customer through fork-join path $i$ is

$$CT_i(\mathbf{n}) = R_{fj}(\mathbf{n})\frac{v^{fj}}{v_1} + \sum_{k \neq fj} R_k(\mathbf{n})\frac{v_k}{v_1} \tag{3.23}$$

where $R_{fj}(\mathbf{n}) = T_i(\mathbf{n})$ and $\frac{v^{fj}}{v_1}$ is the visit ratio to the fork-join queue. The term $\sum_{k \neq fj} R_k(\mathbf{n})\frac{v_k}{v_1}$ is the sum of the response times for the stations outside of the fork-join queue, adjusted by their respective visit ratios.

Before we can determine the average cycle time for the QNM we need one more cycle time, that for the additional subnetwork that bypasses the fork-join queue and consists only of the stations outside of the fork-join queue that do not involve the activities of the fork-join queue. Any station $i$ with transition probability $P_{i,fj} = 1$ to station $fj$, the fork-join queue, is not included in this final subnetwork (for example, in the network illustrated in figure 1.1, station 3, de-arm weapons, is not included in this final cycle time calculation because including it forces the inclusion of the fork-join queue). This final cycle time is

$$CT_{i+1}(\mathbf{n}) = \sum_{k} R_k(\mathbf{n})\frac{v_k}{v_1} \tag{3.24}$$

the sum of the response times for the $k$ stations outside of the fork-join queue that do not involve the activities of the fork-join queue, as described above, adjusted by their respective visit ratios.

In order to determine the QNM's cycle time we average the cycle times calculated for each subnetwork weighted by their respective visit ratios:

$$CT_{QNM}(\mathbf{n}) = \sum_{i=1}^{K} CT_i(\mathbf{n})P_i^{fj}\frac{v^{fj}}{v_1} + CT_{i+1}(\mathbf{n})(1 - \frac{v^{fj}}{v_1}) \tag{3.25}$$

Each $CT_i$ is adjusted by the fork-join node's visit ratio, $\frac{v^{fj}}{v_1}$, and the probability that fork-join queue path $i$ is the one path definitely taken, this being $P_i^{fj}$. Similarly, $CT_{i+1}$ is adjusted by the visit ratio for the subnetwork that bypasses the fork-join queue, which is $(1 - \frac{v^{fj}}{v_1})$. To preclude the possibility of the the term $(1 - \frac{v^{fj}}{v_1})$ being negative, $v_1$ is selected to be a station in the QNM with maximal throughput.

As in Equation 3.7, we determine the throughput through station 1 when $\mathbf{n}$ customers are in the system to be

$$\lambda_1(\mathbf{n}) = \frac{n}{CT_{QNM}(\mathbf{n})} \tag{3.26}$$

For stations outside the fork-join queue (non-maintenance stations), the throughput is applied to Equations 3.8 and 3.9 to determine the the mean queue lengths and server utilizations. However, the throughput may *not* be directly applied to Equations 3.20 and 3.21 to determine the utilization and queue length of each fork-join path queue. We must again account for the probabilistic branching in the fork-join queue. This is accomplished by including terms for the visit ratio to the fork-join queue and the probability of each path being taken. Thus, for each station $i$ in the fork-join queue, Equations 3.20 and 3.21 become:

$$U_i(\mathbf{n}) = s_i \lambda_1(\mathbf{n}) \frac{v^{fj}}{v_1} P_i^{fj} \tag{3.27}$$

and

$$Q_i(\mathbf{n}) = R_i(\mathbf{n}) \lambda_1(\mathbf{n}) \frac{v^{fj}}{v_1} P_i^{fj} \tag{3.28}$$

where $\frac{v^{fj}}{v_1}$ is the visit ratio to the fork-join queue and $P_i^{fj}$ is the probability that fork-join path $i$ is taken. In aircraft sortie generation terms, $\frac{v^{fj}}{v_1}$ is the number of aircraft that experience a system malfunction for every aircraft that taxis, while $P_j^{fj}$ is the probability that system $j$ fails, given that the aircraft has a malfunction.

Once the queue lengths of all network stations when **n** customers in the network are determined, the next iteration of the heuristic can begin. The response time for all stations is updated using the MVA algorithm. Then, the cycle time for each subnetwork followed by the system's cycle time is calculated. The network throughput is determined and finally utilization and queue length for each station is found using the appropriate equations, thus completing the iteration with $\mathbf{n} + 1$ customers in the network. As before, this iterative process begins with $\mathbf{n} = 1$ and ends when $\mathbf{n} = \mathbf{N}$.

### *3.5 Computer Implementation*

The MVA heuristic for aircraft sortie generation was programmed in Borland's Turbo Pascal for Windows, version 1.5. The program calculates and displays the mean values of throughput, queue length, response time and server utilization for each station in the network. The mean values of queue lengths, response time and utilization at each station are displayed for each iteration. The source code for the heuristic, applied to a network with one fork-join queue at station 4, is described and listed in Appendix B. Appendix C contains sample output for the program.

# *IV. Methodology Results and Comparison to LCOM*

## *4.1 Introduction*

This chapter details the results of the MVA heuristic applied to the basic QNM, illustrated in figure 1.1 with figure 1.2 representing the concurrent maintenance activities of station 4. Transition probabilities between queues are shown in the figures. The service time and number of servers for each queue are listed in tables 4.1 and 4.2 below. The infinite number of servers for nonmaintenance stations 1 and 2 indicates these are delay stations.

| Station | 1<br>Taxi | 2<br>Fly | 3<br>De-Arm | 5<br>Turn | 6<br>Re-Arm |
|---|---|---|---|---|---|
| Mean Svc Time (hrs) | 0.25 | 2.00 | 1.25 | 1.15 | 1.25 |
| No. Servers | $\infty$ | $\infty$ | 2 | 8 | 9 |

Table 4.1. Service Time and Number of Servers for Non-Maintenance Stations

| Fork-Join Station | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Mean Svc Time (hrs) | 2.00 | 2.50 | 3.25 | 3.50 | 4.25 | 4.50 |
| No. Servers | 3 | 4 | 4 | 3 | 5 | 5 |

Table 4.2. Service Time and Number of Servers for Fork-Join Queue Stations

After solving the basic QNM, specific network parameters were varied to determine the sensitivity of the heuristic to increased server utilization. At each step, an LCOM simulation of the network was created, the output of which was used as a baseline for determining the accuracy of the heuristic.

## *4.2 LCOM Simulation of the QNM*

Due to the Air Force's acceptance of the Logistics Composite Model's accuracy, LCOM was used to provide the baseline for determining the accuracy of the MVA heuristic for analyzing aircraft sortie generation. The results of the LCOM simulation were taken from the output of the Matrix Post Processor because it is the only source of LCOM data that reports aircraft queueing time at each station. Using this information, plus the number of aircraft served and the manhours worked at each station, network performance characteristics for the simulation were determined.

The basic QNM was coded in LCOM version 93.C format (9) and is contained in Appendix A. As the QNM was modified for each of the sensitivity studies, the LCOM coded version of the network was correspondingly changed. Five iterations of LCOM were run for each network in order to provide confidence limits on the output. Of the 63 simulation output variances calculated for the QNM with 24 aircraft, only six were larger than 0.05, the largest being 0.12. The largest confidence interval, as a percentage of the LCOM output value, corresponded to the largest variance. This occurred for the value of the response time for fork-join path 5 and yielded a 95 percent confidence interval of $4.28 \pm 0.33$ hours. Each run of the LCOM simulation consisted of a 720 hour warmup period followed by 720 hours of data collection.

## *4.3 Results for the basic QNM with 24 Aircraft*

Tables 4.3 and 4.4 below compare the results of the MVA heuristic to the output of the LCOM simulation for the basic QNM with 24 aircraft. For all comparisons to follow, the top line for each station represents the result of the MVA heuristic and the bottom line represents the averaged output of five LCOM runs. The MVA heuristic applied to the non-maintenance stations outside the fork-join queue yielded close agreement with the LCOM simulation output. Stations 1, 2, 5 and 6, exhibit differences across all performance characteristics that range only from 1 to 3 percent. Station 3, the de-arm weapons activity required due to ground abort, had differences

of approximately 10 percent between the heuristic results and simulation output for throughput, queue length, and server utilization. However, the low values of the performance characteristics for this station greatly reduce the significance of the larger percentage difference.

| Station Number | Num of Servers | Mean Svc Time | Visit Ratio | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|---|---|---|
| 1 | $\infty$ | 0.2500 | 1.0000 | 3.5432 | 0.8858 | 0.2500 | 0.8858 |
| | | 0.2481 | 1.0000 | 3.4850 | 0.8647 | 0.2481 | 0.8647 |
| 2 | $\infty$ | 2.0000 | 0.9500 | 3.3661 | 6.7322 | 2.0000 | 6.7322 |
| | | 1.9673 | 0.9528 | 3.3203 | 6.5311 | 1.9673 | 6.5311 |
| 3 | 2 | 1.2500 | 0.0500 | 0.1772 | 0.2239 | 1.2637 | 0.2215 |
| | | 1.2440 | 0.0467 | 0.1628 | 0.2044 | 1.2558 | 0.2025 |
| 5 | 8 | 1.1500 | 1.0000 | 3.5432 | 4.1019 | 1.1577 | 4.0747 |
| | | 1.1374 | 1.0002 | 3.4856 | 3.9894 | 1.1445 | 3.9644 |
| 6 | 9 | 1.2500 | 1.0000 | 3.5432 | 4.4426 | 1.2538 | 4.4291 |
| | | 1.2485 | 0.9998 | 3.4844 | 4.3608 | 1.2514 | 4.3508 |

Table 4.3. Comparison of Heuristic and LCOM Results for Non-Maintenance Stations

While the heuristic result was generally within 5 percent of the simulation output for the maintenance stations on the fork-join queue paths, these differences were less consistent, queue to queue, than for the non-maintenance stations. For example, fork-join queue path 1 had only a 3 percent difference in throughput between the heuristic and simulation, but had a 5 percent difference in response time and a 10 percent difference for queue length and server utilization. On the other hand, fork-join queue path 2 had differences less than 1 percent across all performance characteristics.

### *4.4 Sensitivity Studies*

The purpose of a sensitivity study is to determine the effect of changes to the QNM's parameters (number of customers, number of servers, service time) on

| Station Number | Num of Servers | Mean Svc Time | Visit Ratio | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 2.0000 | 0.1742 | 0.6173 | 1.3159 | 2.1317 | 1.2346 |
| | | 1.9420 | 0.1686 | 0.5878 | 1.1864 | 2.0179 | 1.1417 |
| 2 | 4 | 2.5000 | 0.2043 | 0.7237 | 1.8838 | 2.6029 | 1.8093 |
| | | 2.4972 | 0.2091 | 0.7289 | 1.8900 | 2.5932 | 1.8197 |
| 3 | 4 | 3.2500 | 0.1328 | 0.4707 | 1.5639 | 3.3229 | 1.5296 |
| | | 3.1243 | 0.1357 | 0.4728 | 1.5072 | 3.1886 | 1.4767 |
| 4 | 3 | 3.5000 | 0.1215 | 0.4305 | 1.6862 | 3.9173 | 1.5066 |
| | | 3.5778 | 0.1252 | 0.4358 | 1.7386 | 3.9746 | 1.5619 |
| 5 | 5 | 4.2500 | 0.0988 | 0.3500 | 1.4927 | 4.2644 | 1.4877 |
| | | 4.2657 | 0.1012 | 0.3525 | 1.5120 | 4.2843 | 1.5056 |
| 6 | 5 | 4.5000 | 0.1435 | 0.5085 | 2.3384 | 4.5986 | 2.2882 |
| | | 4.5466 | 0.1478 | 0.5147 | 2.4078 | 4.2843 | 2.3428 |

Table 4.4. Comparison of Heuristic and LCOM Results for Fork-Join Queue

the overall network throughput. Any changes to a specific station's parameters are reflected in that station's server utilization. A decrease in the number of servers or an increase in service time at a station causes its server utilization to increase. Similarly, an increase in the number of servers or a decrease in service time at any station results in lower server utilization at that station. Thus, any sensitity study to changes in the QNM's parameters is equivalent to a sensitivity study to various levels of server utilization.

In order to determine the sensitivity of the MVA heuristic to increased server utilization, the basic QNM with 24 aircraft was re-solved with each of the following changes made:

1. The number of aircraft in the system was doubled from 24 to 48.
2. The number of servers on each of the fork-join queue paths was decreased to one.
3. The number of servers on fork-join path 6 was incrementally reduced.

*4.4.1 48 Aircraft.* The number of aircraft in the system was doubled in order to increase overall congestion, i.e., queueing, at each station. All other parameters were left unchanged. As seen in tables 4.5 and 4.6, while queue lengths and server utilizations nearly doubled, the results of the heuristic continue to be in close agreement with the output of the LCOM simulation for the non-maintenance stations outside the fork-join queue. Differences between simulation and heuristic for these stations, including station 3, are less than 3 percent for all performance characteristics. However, comparison of the results for the multiple-server queues on the fork-join queue paths reveals differences that are again inconsistent in size across the queues and, for this scenario, are much larger. The heuristic result for throughput at fork-join queue path 1 was nearly identical to the simulation output, however, results for queue length, response time and server utilization differed by approximately 19 percent. Fork-join queue path 4 had very small differences for throughput and server utilization, but had differences of approximately 17 percent for queue length and response time. These differences are *not* attributable to the variance of the LCOM simulation output; each of the large errors noted above lies well outside the 95 percent confidence interval for the corresponding LCOM simulation output.

We note that the largest errors occurred on the two fork-join queue paths that have server utilization levels above 80 percent and that, at levels of server utilization 75 percent and below, the MVA heuristic results were within a few percentage points of the LCOM simulation. This leads us to believe that the MVA heuristic loses accurracy for stations on the fork-join queue paths when server utilizations are high.

*4.4.2 Single-Server Queues on Fork-Join Queue Paths.* In an attempt to characterize the levels of server utilization at which the heuristic no longer provides accurate approximations of network performance characteristics, a pathological maintenance scenario was created. Using the basic QNM with 24 aircraft, the number of servers on each of the fork-join queue paths was reduced to one. That is, each of the multiple-server queues was replaced by a single-server queue. Also, in

| Station Number | Num of Servers | Mean Svc Time | Visit Ratio | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|---|---|---|
| 1 | $\infty$ | 0.2500 | 1.0000 | 5.9073 | 1.4768 | 0.2500 | 1.4768 |
| | | 0.2496 | 1.0000 | 6.1286 | 1.5297 | 0.2496 | 1.5297 |
| 2 | $\infty$ | 2.0000 | 0.9500 | 5.6120 | 11.2239 | 2.0000 | 11.2239 |
| | | 2.0024 | 0.9461 | 5.7981 | 11.6106 | 2.0024 | 11.6106 |
| 3 | 2 | 1.2500 | 0.0500 | 0.2954 | 0.3818 | 1.2928 | 0.3692 |
| | | 1.2217 | 0.0509 | 0.3117 | 0.3911 | 1.2573 | 0.3800 |
| 5 | 8 | 1.1500 | 1.0000 | 5.9073 | 8.7918 | 1.4883 | 6.7934 |
| | | 1.1546 | 0.9963 | 6.1058 | 8.7053 | 1.4255 | 7.0503 |
| 6 | 9 | 1.2500 | 1.0000 | 5.9073 | 8.8130 | 1.4919 | 7.3842 |
| | | 1.2390 | 0.9973 | 6.1117 | 8.5608 | 1.4006 | 7.5725 |

Table 4.5. Comparison for Non-Maintenance Stations, 48 Aircraft

| Station Number | Num of Servers | Mean Svc Time | Visit Ratio | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 2.0000 | 0.1742 | 1.0292 | 2.9612 | 2.8773 | 2.0583 |
| | | 1.9634 | 0.1673 | 1.0253 | 2.4781 | 2.4161 | 2.4781 |
| 2 | 4 | 2.5000 | 0.2043 | 1.2066 | 4.3006 | 3.5643 | 2.5502 |
| | | 2.4946 | 0.2020 | 1.2381 | 4.0464 | 3.2673 | 3.0886 |
| 3 | 4 | 3.2500 | 0.1328 | 0.7847 | 3.0706 | 3.9132 | 2.5502 |
| | | 3.2472 | 0.1342 | 0.8222 | 3.1650 | 3.8514 | 2.6675 |
| 4 | 3 | 3.5000 | 0.1215 | 0.7177 | 5.1510 | 7.1776 | 2.5118 |
| | | 3.6108 | 0.1150 | 0.7047 | 4.2189 | 5.9677 | 2.5444 |
| 5 | 5 | 4.2500 | 0.0988 | 0.5836 | 2.5912 | 4.4402 | 2.4802 |
| | | 4.2789 | 0.0981 | 0.6011 | 2.7108 | 4.5148 | 2.5692 |
| 6 | 5 | 4.5000 | 0.1435 | 0.8478 | 5.0409 | 5.9461 | 3.8150 |
| | | 4.3628 | 0.1456 | 0.8919 | 4.9747 | 5.5705 | 3.8906 |

Table 4.6. Comparison for Fork-Join Queue, 48 Aircraft

order to keep the confidence interval on the simulation output reasonably tight, 20 iterations of the LCOM simulation were conducted.

Table 4.7 indicates that the heuristic continued to provide highly accurate results, within 1 to 2 percent of the LCOM simulation, for non-maintenance stations. As shown in table 4.8, the heuristic also accurately calculated throughput and server utilization for the fork-join queue stations. However, for the performance characteristics of queue length and response time, the differences in results increased significantly. The largest differences did not correspond to the highest server utilization, the single-server queue on fork-join queue path 6 with 98 percent server utilization. Instead, the heuristic was most inaccurate for fork-join queue path 4 where results differed from the LCOM simulation by over 50 percent while utilization was only 59 percent. Fork-join queue paths 1, 2, 3 and 5 had differences between the heuristic and simulation that ranged from 21 to 33 percent and server utilizations that ranged from 52 to 78 percent.

The apparent accuracy of the heuristic's results for fork-join path 6 is surprising for two reasons. First, the heuristic's results for queue length and response time are below the values of the LCOM simulation output, which have 95 percent confidence of $11.25 \pm 0.78$ aircraft and $51.95 \pm 4.43$ hours, respectively. For the stations on fork-join queue paths 1 through 5 the heuristic's results for queue length and response time are well above their corresponding LCOM output values. Second, the server utilization on fork-join queue path 6 is 98 percent, by far the highest server utilization, while the heuristic's results for the path are the most accurate. The final sensitivity study will explore the accuracy of the MVA heuristic when the server utilization rate is increased for only one fork-join queue path, path 6 in particular.

*4.4.3 Incremental Reduction of Servers on Fork-Join Path 6.* In order to study the effects of high server utilization on a single fork-join queue path, a final sensitivity study was conducted on the 48 aircraft scenario. The number of servers

| Station Number | Num of Servers | Mean Svc Time | Visit Ratio | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|---|---|---|
| 1 | $\infty$ | 0.2500 | 1.0000 | 1.5114 | 0.3779 | 0.2500 | 0.3779 |
| | | 0.2480 | 1.0000 | 1.5397 | 0.3818 | 0.2480 | 0.3818 |
| 2 | $\infty$ | 2.0000 | 0.9500 | 1.4359 | 2.8717 | 2.0000 | 2.8717 |
| | | 1.9665 | 0.9503 | 1.4631 | 2.8776 | 1.9665 | 2.8776 |
| 3 | 2 | 1.2500 | 0.0500 | 0.0756 | 0.0947 | 1.2528 | 0.0945 |
| | | 1.2385 | 0.0505 | 0.0778 | 0.0963 | 1.2385 | 0.0963 |
| 5 | 8 | 1.1500 | 1.0000 | 1.5114 | 1.7382 | 1.1501 | 1.7381 |
| | | 1.1531 | 1.0009 | 1.5411 | 1.7776 | 1.1531 | 1.7776 |
| 6 | 9 | 1.2500 | 1.0000 | 1.5114 | 1.8893 | 1.2500 | 1.8893 |
| | | 1.2428 | 1.0002 | 1.5401 | 1.9136 | 1.2428 | 1.9136 |

Table 4.7. Comparison for Non-Maintenance Stations, One Server per Fork-Join Path

| Station Number | Num of Servers | Mean Svc Time | Visit Ratio | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 2.0000 | 0.1742 | 0.2633 | 1.1004 | 4.1791 | 0.5266 |
| | | 1.9949 | 0.1714 | 0.2636 | 0.9131 | 3.4596 | 0.5246 |
| 2 | 1 | 2.5000 | 0.2043 | 0.3087 | 3.1037 | 10.0537 | 0.7718 |
| | | 2.4896 | 0.2057 | 0.3164 | 2.5591 | 7.8680 | 0.7892 |
| 3 | 1 | 3.2500 | 0.1328 | 0.2008 | 1.8247 | 9.0887 | 0.6525 |
| | | 3.1482 | 0.1350 | 0.2074 | 1.4227 | 6.8280 | 0.6523 |
| 4 | 1 | 3.5000 | 0.1215 | 0.1836 | 1.7516 | 9.5397 | 0.6427 |
| | | 3.3103 | 0.1164 | 0.1792 | 1.1417 | 6.3453 | 0.5924 |
| 5 | 1 | 4.2500 | 0.0988 | 0.1493 | 1.6943 | 11.3471 | 0.6346 |
| | | 4.2382 | 0.1021 | 0.1569 | 1.4312 | 9.0352 | 0.6652 |
| 6 | 1 | 4.5000 | 0.1435 | 0.2169 | 10.0126 | 46.1607 | 0.9761 |
| | | 4.5469 | 0.1429 | 0.2194 | 11.2474 | 51.9500 | 0.9894 |

Table 4.8. Comparison for Fork-Join Queue, One Server per Path

on fork-join queue path 6 was incrementally reduced by one thereby incrementally increasing server utilization at this station. All other parameters in the network were left unchanged. At every station in the QNM other than the station on fork-join queue path 6, the MVA heuristic accurately determined the network performance characteristics, differing at most by 5 percent from the LCOM simulation output. Table 4.9 displays for fork-join queue path 6 the results of the MVA heuristic and the averaged output of 5 LCOM runs for each level of servers.

| Num of Servers | Thru Put | Queue Length | Response Time | Server Utilization |
|---|---|---|---|---|
| 5 | 0.8478 | 5.0409 | 5.9461 | 3.8150 |
| | 0.8919 | 4.9747 | 5.5705 | 3.8906 |
| 4 | 0.8132 | 8.2908 | 10.2068 | 3.6553 |
| | 0.8322 | 8.4686 | 10.1976 | 3.8622 |
| 3 | 0.6586 | 17.0187 | 25.8409 | 2.9637 |
| | 0.6797 | 17.7133 | 26.2464 | 2.9753 |
| 2 | 0.4412 | 25.2010 | 57.1198 | 1.9854 |
| | 0.4364 | 28.6392 | 65.9732 | 1.9900 |
| 1 | 0.2210 | 32.3064 | 146.2026 | 0.9944 |
| | 0.2110 | 34.3025 | 163.6027 | 0.9875 |

Table 4.9. Comparison for Fork-Join Queue Path 6 Study

MVA heuristic accurately determines the throughput and server utilization as the number of servers decreased. The heuristic's result for response time is 7 percent above the LCOM simulation output when there are 5 servers on the path and server utilization is at 78 percent. As the number of servers decreases and server utilization increases above 95 percent, the heuristic's calculation for the path's response time moves to 13 percent below the LCOM simulation output. The analytical value for queue length follows a similar trending as server utilization increases, reaching a maximum difference of approximately 10 percent. These observations were duplicated for fork-join queue path 4, including the incident of high accuracy at approximately

95 percent utilization. The reasons for this behavior at the very highest levels of server utilization remain unclear.

# V. *Conclusions and Recommendations for Future Research*

## *5.1 General Observations*

This effort was motivated by the Air Force's current policy of using LCOM to determine maintenance manpower requirements. Preparing, running, and optimizing a new Logistics Composite Model simulation is a major undertaking requiring months of effort. System failure data must be collected, detailed task structures must be determined and validated, aircraft mission scenarios and equipment configurations must be identified, LCOM code must be written and debugged and, finally, simulation output must be analyzed. The manpower study itself requires multiple runs of the simulation in order to arrive at the optimum mix of resources to support the desired mission scenario (29).

While the methodology developed in this study is a heuristic, it quickly provides a reasonably accurate estimate of maintenance manpower requirements for a given aircraft sortie generation process. The data required to build the QNM for the heuristic is less complicated than the data used in LCOM. Individual system failure data is not needed; analysts must only determine the probability that each maintenance speciality works on an aircraft that has some malfunction. The average crew size and job completion time for each maintenance speciality replaces detailed task structures. All of this data can be easily determined from the information collected by the Air Force Job Control function that monitors maintenance activities for the actual aircraft sortie generation process to be studied.

The comparisons of the results of the MVA heuristic to the outputs of the LCOM simulation indicate that the heuristic's accuracy decreases as server utilization increases. When server utilization is kept in realistic ranges (utilization of Air Force maintenance personnel rarely exceeds 70 percent (29)), the results of the heuristic for a single fork-join queue are very accurate. For non-maintenance stations, results are within 1 to 2 percent of the LCOM simulation output. For stations

on the fork-join queue paths, heuristic results are within 5 percent of the LCOM simulation's output for that portion of the network.

### *5.2 Limitations of the Methodology*

A key step in the MVA heurisitc for analysis of aircraft sortie generation is the averaging of all of the possible network cycle times to get the actual network cycle time. For our QNM with only one fork-join queue that has 6 paths, this entailed the consideration of only 7 cases. However, if we desire to expand this methodology to complicated QNMs that have more than one fork-join queue, this averaging process could quickly become computationally tedious. For example, a QNM with three fork-join queues with 3,4 and 6 paths would entail the calculation and averaging of 140 different cycle times.

### *5.3 Recommendations for Future Research*

The MVA heuristic developed in this study is a first step towards the creation of an analytical model that accurately relfects a system that has heretofore only been modeled via simulation. Additional work is required to increase the model's fidelity. The topics suggested below are further steps in that direction.

*5.3.1 Heuristic Effect of Assumption (A1).* The methodology developed by Rao and Suri (26) and extended here hinges on the hueristic application of the Arrival Theorem to fork-join queues. Assumption (A1) is exact when there is only one customer in the network and is approximate for more than one customer. The heuristic effect of the assumption and its relation to network parameters such as the number of customers and the number of fork-join queue paths should be investigated. Ideally, this investigation could lead to an adjustment scheme that could be applied to network performance characteristics in order to increase solution accuracy.

*5.3.2 Heuristic Effect of Assumption (A2).* The work presented in this study is also dependent upon Rao and Suri's second assumption, that the response time of an item at fork-join queue path $i$ is exponential. This assumption, like Assumption (A1), is exact only when there is only one customer in the network and is approximate for more than one customer. Making this assumption allows us to apply the memoryless property of the exponential distribution in order to determine the length of time a sibling customer on a fork-join queue path must wait for its other siblings before being re-joined. The actual distribution of the response time is gamma with parameters dependent upon the number of customers in the queue. The gamma distribution has a smaller variance than the exponential, thus the heuristic effect of the assumption is to overestimate the response time. The size of this effect and its relation to network parameters such as the number of customers and the number of fork-join queue paths should be investigated. This investigation could also lead to an adjustment scheme that could be applied to network performance characteristics in order to increase solution accuracy.

*5.3.3 Machine-Operator Interference.* Several simplifying assumptions (detailed in Chapter 1) were made in this study that, while making the QNM easier to model and solve, reduced model accuracy. One simplification was the averaging of crew sizes and service times over each maintenance speciality's tasks. Another dealt with the machine-operator interference caused by separate queues using the same resource. These two issues can be directly addressed without simplification by modeling the aircraft sortie generation process with a hierarchal system of QNMs that exhange information (28:267).

*5.3.4 Validation against an Actual Scenario.* In oder to assess the mathematical accuracy of the heuristic, the validation presented in this study compared the heuristic results to the output of an LCOM simulation of the exact same queueing network model. However, the suitability of using the heuristic as a complement

to LCOM was not addressed. A validation for this purpose would compare an existing LCOM simulation of an actual aircraft sortie generation process, of which many suitable examples exist, to the results of the heuristic applied to an equivalent QNM. Due to the complexity of LCOM tasks structures, this "real-world" validation should be performed as a subset of the machine-operator interference research proposed above.

# Appendix A. *Sample LCOM Input*

```
15
15 FX-999   A              24
15 PILOT1   M              99
15 PILOT2   M              99
15 MAINT1   M               3
15 MAINT2   M               4
15 MAINT3   M               4
15 MAINT4   M               3
15 MAINT5   M               5
15 MAINT6   M               5
15 CHIEF1   M               8
15 WEAPN1   M               9
15 WEAPN2   M               2
25
25 TAXI                 22 0.25H 0.    X          PILOT1   1
25 FLY                  22 2.00H 0.    X          PILOT2   1
25 TASK1                22 2.00H 0.    X          MAINT1   1
25 TASK2                22 2.50H 0.    X          MAINT2   1
25 TASK3                22 3.25H 0.    X          MAINT3   1
25 TASK4                22 3.50H 0.    X          MAINT4   1
25 TASK5                22 4.25H 0.    X          MAINT5   1
25 TASK6                22 4.50H 0.    X          MAINT6   1
25 TURN                 22 1.15H 0.    X          CHIEF1   1
25 ARM                  22 1.25H 0.    X          WEAPN1   1
25 DEARM                22 1.25H 0.    X          WEAPN2   1
25 DUMY                 22             C
30
30 1        TAXI            2       D
30 2        FLY             6       E.95
30 2        DEARM           3       E.05
30 3        CALLUSM         4       C
30 4        TURN            5       D
30 5        ARM             1       D
30 6        DUMY            7       E.65
30 6        TURN            10      E.35
30 10       ARM             1       D
30 7        CALLUSM         8       C
30 8        TURN            9       D
```

```
30 9         ARM                  1         D
30 CALLUSM                        11        A.26
30 11        TASK1                          D
30 CALLUSM                        12        A.31
30 12        TASK2                          D
30 CALLUSM                        13        A.20
30 13        TASK3                          D
30 CALLUSM                        14        A.18
30 14        TASK4                          D
30 CALLUSM                        15        A.15
30 15        TASK5                          D
30 CALLUSM                        16        A.22
30 16        TASK6                          D
45
45 *            24
45   PILOT1     99
45   PILOT2     99              Note:  Each Form 45 has
45   MAINT1      3                     a zero in col 75
45   MAINT2      4
45   MAINT3      4
45   MAINT4      3
45   MAINT5      5
45   MAINT6      5
45   CHIEF1      8
45   WEAPN1      9
45   WEAPN2      2
55
55 ATTACK        A 2                                        FX-999
75
75    1      0001 FX-999   ATTACK   1ALL 0
```

# Appendix B. *Program MVAHEUR*

## *B.1 Description*

The MVA heuristic for analysis of aircraft sortie generation is implemented in the program MVAHEUR. The program is written in Borland's Turbo Pascal for Windows, version 1.5 and is easily changed into standard Pascal format so that it can be run on any mainframe or personal computer with a Pascal compiler. The program, listed below, contains substantial internal documentation including variable definitions and brief descriptions of each subroutine. An overview of the relationship between the various program components is illustrated in figure B.1.

## *B.2 Data Files*

To use program MVAHEUR data should first be entered into a text file called `MVA.DAT` in accordance with the format displayed in Table B.2. The general format is paralleled by the data file for the QNM example with 24 aircraft solved in Chapter 4. In the data file, $N$ is the number of aircraft and $M$ is the number of stations (including the fork-join queue as station 4 but not including the stations on the fork-join paths). $NServer[i]$ is the number of servers at each of these stations with $NServer[4]$ being any value. $S[i]$ is the service time at each of the stations with, again, $S[4]$ being any value. The matrix of transition probabilities, $P[i,j]$, is followed by the fork-join queue parameters for each path. These are the number of servers on each path: $FJServer[i]$, the service time: $FJS[i]$, and the path probabilities: $FJP[i]$. Output of the program is written to a text file called `MVA.OUT`. The output file for the data file below (the QNM with 24 aircraft) is contained in Appendix C.

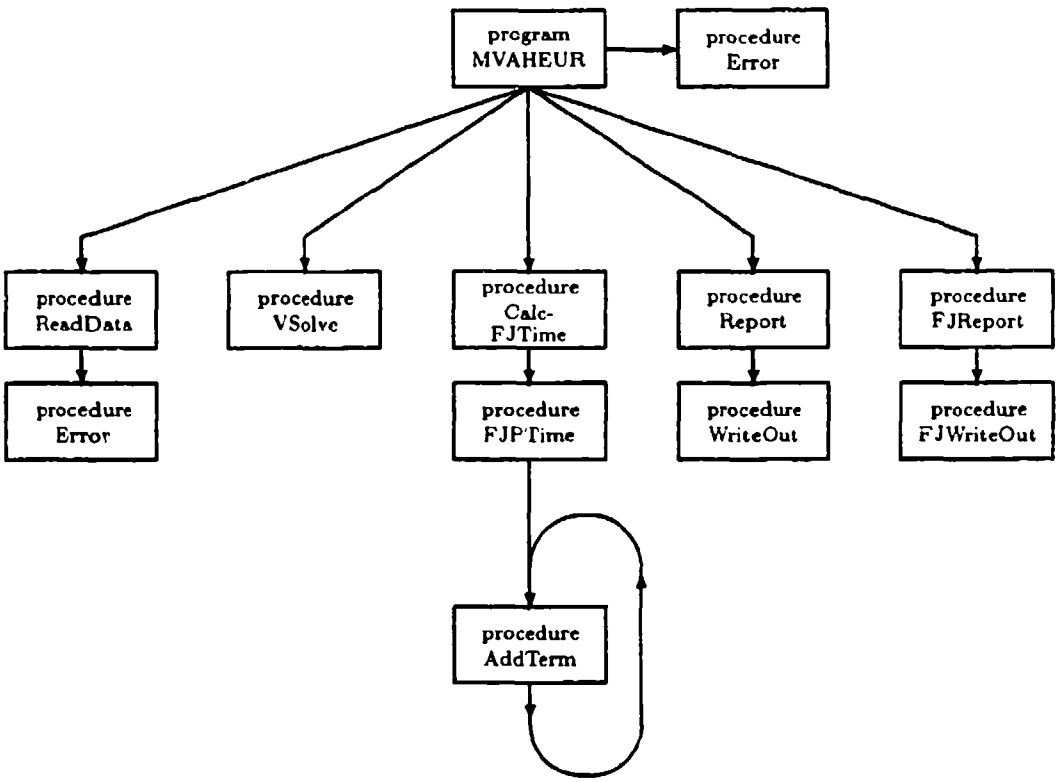



Figure B.1. Relationship Between Components of Program MVAHEUR

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $N$ | $M$ | 24 | 6 | | | | |
| $NServer[i]$ | | 99 | 99 | 2 | 1 | 8 | 9 |
| $S[i]$ | | 0.25 | 2.00 | 1.25 | 1.00 | 1.15 | 1.25 |
| | | 0 | 0.95 | 0.05 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0.65 | 0.35 | 0 |
| $P[i,j]$ | | 0 | 0 | 0 | 1 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 1 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 1 |
| | | 1 | 0 | 0 | 0 | 0 | 0 |
| $FJServer$ | | 3 | 4 | 4 | 3 | 5 | 5 |
| $FJS[i]$ | | 2.00 | 2.50 | 3.25 | 3.50 | 4.25 | 4.50 |
| $FJP[i]$ | | 0.261 | 0.306 | 0.199 | 0.182 | 0.148 | 0.215 |

Table B.1. Data Format for Program MVAHEUR

## B.3 Program Listing

```
program MVAHEUR;

{*************************************************************}
{*                                                           *}
{*  Applies a Mean Value Analysis Hueristic to determine     *}
{*  performance measures for stations in a closed network    *}
{*  of multiserver queues where one queue is a Fork-Join     *}
{*  Queue                                                    *}
{*                                                           *}
{*  Language:  Turbo-Pascal for Windows 1.5                  *}
{*                                                           *}
{*  Source:  R C Jenkins, rjenkins@afit.af.mil, Box 4498     *}
{*           AFIT/ENS, Wright-Patterson AFB OH 45433-7765    *}
{*                                                           *}
{*************************************************************}
uses WinCrt;

const
      Nmax = 48; {maximum number of aircraft in network}
      Mmax = 9; {maximum number of stations in network}
      Paths = 6; {maximum number of paths in the FJ queue}

type Age = (old,new);
     MIntArray = array[1..Mmax] of integer;
     MRealArray = array[1..Mmax] of real;
     PMatrix = array[1..Mmax,1..Mmax+1] of real;
     FJIntArray = array[1..Paths] of integer;
     FJRealArray = array[1..Paths] of real;
     OutMatrix = array[1..Mmax,0..Nmax] of real;
     FJOutMatrix = array[1..Paths,0..Nmax] of real;
     NRealArray = array[1..Nmax] of real;
     FJTimeArray = array[0..Paths] of real;
     FJTimeArray2 = array[0..Paths+1] of real;
     SubPrArray = array[1..Paths+1] of real;
     CTMatrix = array[1..Paths+1,0..Mmax] of real;

var I,J,K,FJ: integer; {counters}
    M,N: integer;      {number of stations, customers in QNM}
    CycleTime: real;   {cycletime of QNM}
    Lambda: NRealArray; {throughput of QNM}
    MServer: MIntArray; {number of servers at each station}
    FJServer: FJIntArray; {number of servers at fork-join queue path}
    S: MRealArray;      {service time of stations}
    V: MRealArray;      {visit ratios of stations}
    FJS: FJRealArray;   {service time of fork-join queue path stations}
    FJP: FJRealArray;   {fork-join queue path probabilities}
    P: PMatrix;         {transition probability matrix for QNM}
    C: array[1..Mmax,1..Nmax] of integer; {number of busy servers per station}
    FJC: array[1..Paths,1..Nmax] of real; {number of busy servers on fj paths}
```

```
    Q,R,U: OutMatrix; {output matrices for non-maintenance stations}
    FJR,FJQ,FJU: FJOutMatrix;  {Output data matrices for fork-join queue}
    Pr: array[old..new] of OutMatrix; {probabilities for marginal local...}
    FJPr: array[old..new]of FJOutMatrix; {...balance theorem application}
    Dat,Out: text;      {input and output files}

{************************************************************}

procedure Error(ErrCode,I,K: integer);

{Reports errors - called by program MVAHEUR and procedure ReadData}

begin
    writeln;
    case ErrCode of
        1: writeln('ERROR: N not in {1..',Nmax:2,'}');
        2: writeln('ERROR: M not in {1..',Mmax:2,'}');
        3: writeln('ERROR: routing probs from station ',I:2,
             ' do not sum to 1.0000');
        4: writeln('WARNING: Pr{',I:2,',0,',K:2,'}<0; ',
             'may have numerical problems');
        5: writeln('FJWARNING: Pr{',I:2,',0,',K:2,'}<0; ',
             'may have numerical problems');
      end; {case}
    if (ErrCode=4 or 5) then begin
        writeln('Press <enter> to resume');
        readln;
      end else begin
        writeln('Program terminated; press <enter> to exit');
        readln;
        halt;
      end; {else}
  end; {Error}
```

```
{*********************************************************}

procedure ReadData(var N,M: integer;
                   var NServer: MIntArray;
                   var FJServer: FJIntArray;
                   var S: MRealArray;
                   var FJS,FJP:FJRealArray;
                   var P: PMatrix;
                   var Dat: text);

{Reads input data from file assigned to text variable 'Dat'}
{Called by program MVAHEUR}

var I,J: integer; {counters}
    Psum: real;   {holds sum of transition probabilities matrix rows}

begin
    readln(Dat,N,M);
    if not (N in [1..Nmax]) then Error(1,0,0);
    if not (M in [1..Mmax]) then Error(2,0,0);
    for I:=1 to M do read(Dat,NServer[I]);
    for I:=1 to M do read(Dat,S[I]);
    for I:=1 to M do begin
        Psum:=0;
        for J:=1 to M do begin
            read(Dat,P[J,I]);
            Psum:=Psum+P[J,I];
          end; {for}
        if (Psum<0.9999) or (Psum>1.0001) then Error(3,I,0);
      end; {for}
    for I:= 1 to Paths do read(Dat,FJServer[I]);
    for I:= 1 to Paths do read(Dat,FJS[I]);
    for I:= 1 to Paths do read(Dat,FJP[I]);
    P[1,1]:=1;
    for J:=2 to M do P[1,J]:=0;
    P[1,M+1]:=1;
    for I:=2 to M do begin
        P[I,I]:=P[I,I]-1;
        P[I,M+1]:=0;
      end; {for}
  end; {ReadData}
```

```
{*********************************************************}

procedure VSolve(M: integer;
                 var P: PMatrix;
                 var V: MRealArray);

{Solves simultaneous equations to obtain visit ratios}
{Called by MVAHEUR}

var I,J,K,IC,KK,MM,IT,IS: integer;  {counters}
    B,W,C: real;
    ID: MIntArray;
    Y: MRealArray;

begin
    MM:=M+1;
    for I:=1 to M do ID[I]:=I;
    K:=1;
    repeat
        KK:=K+1;
        IS:=K;
        IT:=K;
        B:=abs(P[K,K]);
        for I:=K to M do for J:=K to M do
          if (abs(P[I,J])>B) then begin
              IS:=I;
              IT:=J;
              B:=abs(P[I,J]);
            end; {if}
        if (IS>K) then
          for J:=K to MM do begin
              C:=P[IS,J];
              P[IS,J]:=P[K,J];
              P[K,J]:=C;
            end; {for}
        if (IT>K) then begin
            IC:=ID[K];
            ID[K]:=ID[IT];
            ID[IT]:=IC;
            for I:=1 to M do begin
                C:=P[I,IT];
                P[I,IT]:=P[I,K];
                P[I,K]:=C;
              end; {for}
          end; {if}
        for J:=KK to MM do begin
            P[K,J]:=P[K,J]/P[K,K];
            for I:=KK to M do begin
                W:=P[I,K]*P[K,J];
                P[I,J]:=P[I,J]-W;
```

```
            if (abs(P[I,J])<0.00001*abs(W)) then P[I,J]:=0;
          end; {for}
        end; {for}
      K:=KK;
    until (K=M);
  Y[M]:=P[M,MM]/P[M,M];
  for I:=1 to M-1 do begin
      K:=M-I;
      KK:=K+1;
      Y[K]:=P[K,MM];
      for J:=KK to M do Y[K]:=Y[K]-P[K,J]*Y[J];
    end; {for}
  for I:=1 to M do for J:=1 to M do
    if (ID[J]=I) then V[I]:=Y[J];
end; {VSolve}
```

```
{*************************************************************}

procedure AddTerm(J,sign: integer;
                  num,den:real;
                  var I: integer;
                  var pr: real;
                  var T: FJRealArray);

{calculates terms for fork-join queue waiting probabiities}
{Called by procedure FJPTime and calls itself}

var K: integer; {counter}

begin
    num:=num+T[J];
    den:=den+T[J];
    sign:=sign*(-1);
    pr:=pr+sign*num/den;
    if (J<I-1) then
      for K:=J+1 to I-1 do AddTerm(K,sign,num,den,I,pr,T);
  end; {AddTerm}

{*************************************************************}

procedure FJPTime ( Paths: Integer;
                    T: FJRealArray;
                    FJR: FJOutmatrix;
                    FJP: FJRealArray;
                    var Time: Real);

{determines response time of fork-join queue for each path}
{Called by CalcCycleTime, calls AddTerm}

var FJTPr,Term: FJRealArray;
    WTime: FJTimeArray;
    sign,I,J: integer;
    num,den,pr: real;

begin
    for I:=1 to Paths do begin
        pr:=0;
        num:=0;
        den:=T[I];
        for J:=1 to I-1 do begin
            sign:=-1;
            AddTerm(J,sign,num,den,I,pr,T);
          end; {for}
        FJTPr[I]:=pr;
      end; {for}
```

```
  for I:=1 to Paths do Term[I]:=FJTPr[I]*FJR[I,K]*FJP[I];
  WTime[0]:=0;
  for I:= 1 to Paths do WTime[I]:=WTime[I-1]+Term[I];
  Time:= FJR[1,K] + WTime[6];
end; {FJTime}
```

```
{*************************************************************}

procedure CalcCycleTime(Paths,M: Integer;
                        V: MRealArray;
                        FJR: FJOutmatrix;
                        FJP: FJRealArray;
                        R: OutMatrix;
                        var CycleTime: Real);

{averages fork-join response times for all paths}
{Called by program MVAHEUR, calls FJPTime}

var PTime: FJRealArray; {response time of subnetworks}
    T: FJRealArray; {inverse fork-join queue response times}
    TotProb: FJTimeArray;  {sums of subnetwork probabilities}
    TotTime: FJTimeArray2; {sums of cycletimes adjusted by probs}
    I,J: integer;  {counters}
    Time: Real;  {response time of fork-join queue}
    CT: CTMatrix; {cycletime for each subnetwork}
    SubPr: SubPrArray; {probability of a subnetwork}

begin
    for I:= 1 to Paths do begin
        T[I]:=1/FJR[I,K];
      end; {for}
    FJPTime(Paths,T,FJR,FJP,Time);
    PTime[1]:=Time;

    for J:=1 to 5 do begin
        for I:=1 to 5 do begin
            T[I]:=T[I+1];
            FJR[I,K]:=FJR[I+1,K];
            FJP[I]:=FJP[I+1];
          end; {for}
        T[6]:=T[1];
        FJR[6,K]:=FJR[1,K];
        FJP[6]:=FJP[1];
        FJPTime(Paths,T,FJR,FJP,Time);
        PTime[J+1]:=Time;
      end;{for}

    For I:=1 to Paths do begin
        R[4,K]:=PTime[I];
        CT[I,0]:=0;
        for J:=1 to M do CT[I,J]:=CT[I,J-1]+V[J]*R[J,K];
      end; {for}

    CT[7,M]:=V[1]*R[1,K]+V[2]*R[2,K]+V[5]*R[5,K]+V[6]*R[6,K];

    TotProb[0]:=0;
```

```
  for I:=1 to Paths do begin
      TotProb[I]:= TotProb[I-1]+FJP[I]*V[4];
    end; {for}

  for I:=1 to Paths do SubPr[I]:=FJP[I]*V[4];
  SubPr[7]:=1-V[4];

  TotTime[0]:=0;
  for I:=1 to Paths+1 do begin
      TotTime[I]:=Tottime[I-1]+CT[I,M]*SubPr[I]/(TotProb[6]+(1-V[4]));
    end; {for}
  CycleTime:=TotTime[7];
end; {CalcFJTime}
```

```
{************************************************************}

procedure WriteOut(N,M: integer;
                   var X: OutMatrix;
                   var Out: text);

{Writes complete output matrix for performance measures}
{Called by procedure Report}

var I,K: integer; {counters}

begin
    writeln(Out);
    write(Out,'  N');
    for I:=1 to M do write(Out,'   i=',I:2);
    writeln(Out);
    write(Out,' --');
    for I:=1 to M do write(Out,'  -----');
    writeln(Out);
    for K:=1 to N do begin
        write(Out,K:3);
        for I:=1 to M do write(Out,X[I,K]:7:3);
        writeln(Out);
      end; {for}
    writeln(Out);
    writeln(Out);
  end; {WriteOut}

{************************************************************}

procedure FJWriteOut(N,Paths: integer;
                   var FJX: FJOutMatrix;
                   var Out: text);

{Writes output for Fork-Join Queue performance measures}
{Called by procedure FJReport}

var I,K: integer; {counters}

begin
    writeln(Out);
    write(Out,'  N');
    for I:=1 to Paths do write(Out,'   i=',I:2);
    writeln(Out);
    write(Out,' --');
    for I:=1 to Paths do write(Out,'  -----');
    writeln(Out);
    for K:=1 to N do begin
        write(Out,K:3);
        for I:=1 to Paths do write(Out,FJX[I,K]:7:3);
```

```
        writeln(Out);
      end; {for}
    writeln(Out);
    writeln(Out);
end; {FJWriteOut}
```

```
{************************************************************}

procedure Report(N,M: integer;
                 Lambda: NRealArray;
                 var NServer: MIntArray;
                 var S,V: MRealArray;
                 var Q,R,U: OutMatrix;
                 var Out: text);

{Writes output report to file assigned to text variable 'Out'}
{Called by MVAHEUR, calls procedure WriteOut}

var I: integer; {counter}
    SLambda: real; {individual station's throughput}

begin
    writeln(Out,'MVA PERFORMANCE REPORT');
    writeln(Out);
    writeln(Out,'Number of Customers (N) =',N:3);
    writeln(Out);
    writeln(Out,'Stat  Num of   Mean    Visit   Thru ',
      '  Queue   Respns  Server');
    writeln(Out,' (i)  Servrs  Svc Tm   Ratio    put ',
      '  Length   Time    Util ');
    writeln(Out,'----  ------  ------  ------  ------',
      '  ------  ------  ------');
    for I:=1 to M do begin
        SLambda:=Lambda[N]*V[I];
        writeln(Out,I:3,NServer[I]:7,S[I]:10:4,V[I]:8:4,
          SLambda:8:4,Q[I,N]:8:4,R[I,N]:8:4,U[I,N]:8:4);
      end; {for}
    writeln(Out);
    writeln(Out);
    writeln(Out,'Average Queue Lengths (including service)');
    WriteOut(N,M,Q,Out);
    writeln(Out,'Average Response Times (including service)');
    WriteOut(N,M,R,Out);
    Writeln(Out,'Average Utilizations');
    WriteOut(N,M,U,Out);
  end; {Report}
```

```
{*********************************************************}

procedure FJReport(N,M,Paths: integer;
                   V: MRealArray;
                   Lambda: NRealArray;
                   var FJServer: FJIntArray;
                   var FJS,FJP: FJRealArray;
                   var FJQ,FJR,FJU: FJOutMatrix;
                   var R: OutMatrix;
                   var Out: text);

{Writes output for FJ queue to file assigned text variable 'Out'}
{Called by MVAHEUR, calls FJWriteOut}

var I: integer; {counter}
    FJLambda: real; {individual fork-join path throughput}

begin
    writeln(Jut,'MVA PERFORMANCE REPORT FOR FORK-JOIN QUEUE');
    writeln(Out);
    writeln(Out,'Number of Customers (N) =',N:3);
    writeln(Out);
    writeln(Out,'Stat  Num of   Mean   Visit    Thru ',
      '  Queue   Respns  Server');
    writeln(Out,' (i)  Servrs  Svc Tm  Ratio     put ',
      '  Length   Time    Util ');
    writeln(Out,'----  ------  ------  ------  ------',
      '  ------  ------  ------');
    for I:=1 to Paths do begin
        FJLambda:=Lambda[N]*FJP[I]*V[4];
        writeln(Out,I:3,FJServer[I]:7,FJS[I]:10:4, V[4]*FJP[I]:8:4,
          FJLambda:8:4,FJQ[I,N]:8:4,FJR[I,N]:8:4,FJU[I,N]:8:4);
      end; {for}
    writeln(Out);
    writeln(Out);
    writeln(Out,'Average Fork-Join Queue Lengths (including service)');
    FJWriteOut(N,Paths,FJQ,Out);
    writeln(Out,'Average Response Times (including service)');
    FJWriteOut(N,Paths,FJR,Out);
    Writeln(Out,'Average Utilizations');
    FJWriteOut(N,Paths,FJU,Out);
  end; {FJReport}
```

```
{*********************************************************}

begin
    assign(Dat,'MVA.DAT');
    assign(Out,'MVA.OUT');
    reset(Dat);
    rewrite(Out);
    writeln;
    writeln('*** Running program MVA ***');
    writeln;
    writeln('> Reading data ...');
    ReadData(N,M,NServer,FJServer,S,FJS,FJP,P,Dat);
    writeln('> Calculating visit ratios ...');
    VSolve(M,P,V);
    writeln('> Calculating performance measures ...');
    for I:=1 to M do begin;
        for K:=1 to N do
          if (K<NServer[I]) then C[I,K]:=K else C[I,K]:=NServer[I];
        Pr[old,I,0]:=1;
        Q[I,0]:=0;
      end; {for}
    for I:=1 to Paths do begin
        for K:=1 to N do begin
            if (K<FJServer[I]) then FJC[I,K]:=K else
              FJC[I,K]:=FJServer[I];
            FJPr[old,I,0]:=1;
            FJQ[I,0]:=0;
          end; {for}
      end; {for}
   for K:=1 to N do begin
        for I:=1 to 3 do begin
            if (NServer[I]>=N) then R[I,K]:=S[I] else
                if (NServer[I]=1) then
                  R[I,K]:=S[I]*(1+Q[I,K-1]) else begin
                      R[I,K]:=0;
                      for J:=1 to K do
                        R[I,K]:=R[I,K]+J*Pr[old,I,J-1]/C[I,J];
                      R[I,K]:=R[I,K]*S[I];
                    end; {else}
          end; {for}
        for I:= 1 to Paths do begin
            if (FJServer[I]>=V[4]*N) then FJR[I,K]:=FJS[I];
                if (FJServer[I] = 1) then
                   FJR[I,K]:=FJS[I]*(1 + FJQ[I,K-1]) else begin
                       FJR[I,K]:=0;
                       For J:= 1 to K do
                         FJR[I,K]:=FJR[I,K] + J*FJPr[old,I,J-1]/FJC[I,J];
                       FJR[I,K]:=FJR[I,K]*FJS[I];
                     end; {else}
          end; {for}
```

```
    for I:=5 to M do begin
        if (MServer[I]>=N) then R[I,K]:=S[I] else
            if (MServer[I]=1) then
              R[I,K]:=S[I]*(1+Q[I,K-1]) else begin
                  R[I,K]:=0;
                  for J:=1 to K do
                    R[I,K]:=R[I,K]+J*Pr[old,I,J-1]/C[I,J];
                  R[I,K]:=R[I,K]*S[I];
                end; {else}
      end; {for}
    CalcCycleTime(Paths,M,V,FJR,FJP,R,CycleTime);
    Lambda[K]:=K/CycleTime;
    for I:=1 to M do begin
        Q[I,K]:=R[I,K]*Lambda[K]*V[I];
        U[I,K]:=S[I]*Lambda[K]*V[I];
      end; {for}
    for I:= 1 to Paths do begin
        FJQ[I,K]:=FJR[I,K]*Lambda[K]*FJP[I]*V[4];
        FJU[I,K]:=FJS[I]*Lambda[K]*FJP[I]*V[4];
      end; {for}
    for I:=1 to 3 do if (MServer[I]<N) then begin
        for J:=1 to K do Pr[new,I,J]:=U[I,K]*Pr[old,I,J-1]/C[I,J];
        Pr[new,I,0]:=1;
        for J:=1 to K do Pr[new,I,0]:=Pr[new,I,0]-Pr[new,I,J];
        if (Pr[new,I,0]<0) then Error(4,I,K);
      end; {for}
    Pr[old]:=Pr[new];
    for I:=1 to Paths do if (FJServer[I]<V[4]*N) then begin
        for J:=1 to K do FJPr[new,I,J]:=FJU[I,K]*
        FJPr[old,I,J-1]/FJC[I,J];
        FJPr[new,I,0]:=1;
        for J:=1 to K do FJPr[new,I,0]:=FJPr[new,I,0]-FJPr[new,I,J];
        if (FJPr[new,I,0]<0) then Error(5,I,K);
      end; {for}
    FJPr[old]:=FJPr[new];
    for I:=5 to M do if (MServer[I]<N) then begin
        for J:=1 to K do Pr[new,I,J]:=U[I,K]*Pr[old,I,J-1]/C[I,J];
        Pr[new,I,0]:=1;
        for J:=1 to K do Pr[new,I,0]:=Pr[new,I,0]-Pr[new,I,J];
        if (Pr[new,I,0]<0) then Error(4,I,K);
      end; {for}
    Pr[old]:=Pr[new];
  end;
writeln('> Writing output ...');
Report(N,M,Lambda,MServer,S,V,Q,R,U,Out);
FJReport(N,M,Paths,V,Lambda,FJServer,FJS,FJP,FJQ,FJR,FJU,R,Out);
close(Dat);
close(Out);
writeln;
writeln('Program complete; output to file MVA.OUT');
writeln;
```

```
end. {MVA}
```

# Appendix C. *Sample Output*

MVA PERFORMANCE REPORT

Number of Customers (N) = 24

| Stat (i) | Num of Servrs | Mean Svc Tm | Visit Ratio | Thru put | Queue Length | Respns Time | Server Util |
|---|---|---|---|---|---|---|---|
| 1 | 99 | 0.2500 | 1.0000 | 3.5432 | 0.8858 | 0.2500 | 0.8858 |
| 2 | 99 | 2.0000 | 0.9500 | 3.3661 | 6.7322 | 2.0000 | 6.7322 |
| 3 | 2 | 1.2500 | 0.0500 | 0.1772 | 0.2239 | 1.2637 | 0.2215 |
| 4 | 1 | 1.0000 | 0.6675 | 2.3651 | 0.0000 | 0.0000 | 2.3651 |
| 5 | 8 | 1.1500 | 1.0000 | 3.5432 | 4.1019 | 1.1577 | 4.0747 |
| 6 | 9 | 1.2500 | 1.0000 | 3.5432 | 4.4426 | 1.2538 | 4.4291 |

Average Queue Lengths (including service)

| N | i= 1 | i= 2 | i= 3 | i= 4 | i= 5 | i= 6 |
|---|---|---|---|---|---|---|
| 1 | 0.037 | 0.285 | 0.009 | 0.000 | 0.172 | 0.187 |
| 2 | 0.075 | 0.569 | 0.019 | 0.000 | 0.345 | 0.374 |
| 3 | 0.112 | 0.854 | 0.028 | 0.000 | 0.517 | 0.562 |
| 4 | 0.150 | 1.138 | 0.037 | 0.000 | 0.689 | 0.749 |
| 5 | 0.187 | 1.423 | 0.047 | 0.000 | 0.861 | 0.936 |
| 6 | 0.225 | 1.708 | 0.056 | 0.000 | 1.034 | 1.123 |
| 7 | 0.262 | 1.992 | 0.066 | 0.000 | 1.206 | 1.311 |
| 8 | 0.300 | 2.276 | 0.075 | 0.000 | 1.378 | 1.498 |
| 9 | 0.337 | 2.561 | 0.084 | 0.000 | 1.550 | 1.685 |
| 10 | 0.374 | 2.845 | 0.094 | 0.000 | 1.722 | 1.871 |
| 11 | 0.412 | 3.128 | 0.103 | 0.000 | 1.894 | 2.058 |
| 12 | 0.449 | 3.412 | 0.113 | 0.000 | 2.065 | 2.245 |
| 13 | 0.486 | 3.695 | 0.122 | 0.000 | 2.236 | 2.431 |
| 14 | 0.523 | 3.977 | 0.131 | 0.000 | 2.407 | 2.617 |
| 15 | 0.560 | 4.259 | 0.141 | 0.000 | 2.578 | 2.802 |
| 16 | 0.597 | 4.540 | 0.150 | 0.000 | 2.749 | 2.987 |
| 17 | 0.634 | 4.820 | 0.159 | 0.000 | 2.919 | 3.172 |
| 18 | 0.671 | 5.099 | 0.169 | 0.000 | 3.088 | 3.355 |
| 19 | 0.707 | 5.376 | 0.178 | 0.000 | 3.258 | 3.538 |
| 20 | 0.744 | 5.652 | 0.187 | 0.000 | 3.427 | 3.721 |
| 21 | 0.780 | 5.926 | 0.197 | 0.000 | 3.596 | 3.902 |
| 22 | 0.815 | 6.197 | 0.206 | 0.000 | 3.764 | 4.083 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 23 | 0.851 | 6.466 | 0.215 | 0.000 | 3.933 | 4.263 |
| 24 | 0.886 | 6.732 | 0.224 | 0.000 | 4.102 | 4.443 |

Average Response Times (including service)

| N | i= 1 | i= 2 | i= 3 | i= 4 | i= 5 | i= 6 |
|---|---|---|---|---|---|---|
| 1 | 0.250 | 2.000 | 1.250 | 0.000 | 1.150 | 1.250 |
| 2 | 0.250 | 2.000 | 1.250 | 0.000 | 1.150 | 1.250 |
| 3 | 0.250 | 2.000 | 1.250 | 0.000 | 1.150 | 1.250 |
| 4 | 0.250 | 2.000 | 1.250 | 0.000 | 1.150 | 1.250 |
| 5 | 0.250 | 2.000 | 1.250 | 0.000 | 1.150 | 1.250 |
| 6 | 0.250 | 2.000 | 1.251 | 0.000 | 1.150 | 1.250 |
| 7 | 0.250 | 2.000 | 1.251 | 0.000 | 1.150 | 1.250 |
| 8 | 0.250 | 2.000 | 1.251 | 0.000 | 1.150 | 1.250 |
| 9 | 0.250 | 2.000 | 1.252 | 0.000 | 1.150 | 1.250 |
| 10 | 0.250 | 2.000 | 1.252 | 0.000 | 1.150 | 1.250 |
| 11 | 0.250 | 2.000 | 1.252 | 0.000 | 1.150 | 1.250 |
| 12 | 0.250 | 2.000 | 1.253 | 0.000 | 1.150 | 1.250 |
| 13 | 0.250 | 2.000 | 1.254 | 0.000 | 1.150 | 1.250 |
| 14 | 0.250 | 2.000 | 1.254 | 0.000 | 1.150 | 1.250 |
| 15 | 0.250 | 2.000 | 1.255 | 0.000 | 1.150 | 1.250 |
| 16 | 0.250 | 2.000 | 1.256 | 0.000 | 1.150 | 1.250 |
| 17 | 0.250 | 2.000 | 1.257 | 0.000 | 1.150 | 1.250 |
| 18 | 0.250 | 2.000 | 1.257 | 0.000 | 1.151 | 1.250 |
| 19 | 0.250 | 2.000 | 1.258 | 0.000 | 1.151 | 1.250 |
| 20 | 0.250 | 2.000 | 1.259 | 0.000 | 1.152 | 1.251 |
| 21 | 0.250 | 2.000 | 1.260 | 0.000 | 1.153 | 1.251 |
| 22 | 0.250 | 2.000 | 1.261 | 0.000 | 1.154 | 1.252 |
| 23 | 0.250 | 2.000 | 1.263 | 0.000 | 1.156 | 1.253 |
| 24 | 0.250 | 2.000 | 1.264 | 0.000 | 1.158 | 1.254 |

Average Utilizations

| N | i= 1 | i= 2 | i= 3 | i= 4 | i= 5 | i= 6 |
|---|---|---|---|---|---|---|
| 1 | 0.037 | 0.285 | 0.009 | 0.100 | 0.172 | 0.187 |
| 2 | 0.075 | 0.569 | 0.019 | 0.200 | 0.345 | 0.374 |
| 3 | 0.112 | 0.854 | 0.028 | 0.300 | 0.517 | 0.562 |
| 4 | 0.150 | 1.138 | 0.037 | 0.400 | 0.689 | 0.749 |
| 5 | 0.187 | 1.423 | 0.047 | 0.500 | 0.861 | 0.936 |
| 6 | 0.225 | 1.708 | 0.056 | 0.600 | 1.034 | 1.123 |
| 7 | 0.262 | 1.992 | 0.066 | 0.700 | 1.206 | 1.311 |
| 8 | 0.300 | 2.276 | 0.075 | 0.800 | 1.378 | 1.498 |
| 9 | 0.337 | 2.561 | 0.084 | 0.900 | 1.550 | 1.685 |
| 10 | 0.374 | 2.845 | 0.094 | 0.999 | 1.722 | 1.871 |
| 11 | 0.412 | 3.128 | 0.103 | 1.099 | 1.894 | 2.058 |
| 12 | 0.449 | 3.412 | 0.112 | 1.199 | 2.065 | 2.245 |
| 13 | 0.486 | 3.695 | 0.122 | 1.298 | 2.236 | 2.431 |
| 14 | 0.523 | 3.977 | 0.131 | 1.397 | 2.407 | 2.617 |
| 15 | 0.560 | 4.259 | 0.140 | 1.496 | 2.578 | 2.802 |
| 16 | 0.597 | 4.540 | 0.149 | 1.595 | 2.748 | 2.987 |
| 17 | 0.634 | 4.820 | 0.159 | 1.693 | 2.917 | 3.171 |
| 18 | 0.671 | 5.099 | 0.168 | 1.791 | 3.086 | 3.355 |
| 19 | 0.707 | 5.376 | 0.177 | 1.889 | 3.254 | 3.537 |
| 20 | 0.744 | 5.652 | 0.186 | 1.986 | 3.421 | 3.719 |
| 21 | 0.780 | 5.926 | 0.195 | 2.082 | 3.587 | 3.899 |
| 22 | 0.815 | 6.197 | 0.204 | 2.177 | 3.751 | 4.077 |
| 23 | 0.851 | 6.466 | 0.213 | 2.272 | 3.914 | 4.254 |
| 24 | 0.886 | 6.732 | 0.221 | 2.365 | 4.075 | 4.429 |

MVA PERFORMANCE REPORT FOR FORK-JOIN QUEUE

Number of Customers (N) = 24

| Stat (i) | Num of Servrs | Mean Svc Tm | Visit Ratio | Thru put | Queue Length | Respns Time | Server Util |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 2.0000 | 0.1742 | 0.6173 | 1.3159 | 2.1317 | 1.2346 |
| 2 | 4 | 2.5000 | 0.2043 | 0.7237 | 1.8838 | 2.6029 | 1.8093 |
| 3 | 4 | 3.2500 | 0.1328 | 0.4707 | 1.5639 | 3.3229 | 1.5296 |
| 4 | 3 | 3.5000 | 0.1215 | 0.4305 | 1.6862 | 3.9173 | 1.5066 |
| 5 | 5 | 4.2500 | 0.0988 | 0.3500 | 1.4927 | 4.2644 | 1.4877 |
| 6 | 5 | 4.5000 | 0.1435 | 0.5085 | 2.3384 | 4.5986 | 2.2882 |

Average Fork-Join Queue Lengths (including service)

| N | i= 1 | i= 2 | i= 3 | i= 4 | i= 5 | i= 6 |
|---|---|---|---|---|---|---|
| 1 | 0.052 | 0.076 | 0.065 | 0.064 | 0.063 | 0.097 |
| 2 | 0.104 | 0.153 | 0.129 | 0.127 | 0.126 | 0.193 |
| 3 | 0.157 | 0.229 | 0.194 | 0.191 | 0.189 | 0.290 |
| 4 | 0.209 | 0.306 | 0.259 | 0.255 | 0.252 | 0.387 |
| 5 | 0.261 | 0.382 | 0.323 | 0.319 | 0.314 | 0.484 |
| 6 | 0.313 | 0.459 | 0.388 | 0.382 | 0.377 | 0.580 |
| 7 | 0.366 | 0.535 | 0.453 | 0.447 | 0.440 | 0.677 |
| 8 | 0.413 | 0.612 | 0.517 | 0.511 | 0.503 | 0.774 |
| 9 | 0.471 | 0.689 | 0.582 | 0.576 | 0.566 | 0.870 |
| 10 | 0.524 | 0.765 | 0.647 | 0.641 | 0.629 | 0.967 |
| 11 | 0.577 | 0.842 | 0.711 | 0.706 | 0.691 | 1.064 |
| 12 | 0.630 | 0.919 | 0.776 | 0.773 | 0.754 | 1.160 |
| 13 | 0.684 | 0.996 | 0.841 | 0.840 | 0.817 | 1.257 |
| 14 | 0.738 | 1.074 | 0.906 | 0.909 | 0.879 | 1.354 |
| 15 | 0.793 | 1.152 | 0.971 | 0.978 | 0.941 | 1.451 |
| 16 | 0.848 | 1.230 | 1.036 | 1.049 | 1.004 | 1.548 |
| 17 | 0.904 | 1.309 | 1.101 | 1.121 | 1.066 | 1.645 |
| 18 | 0.960 | 1.388 | 1.167 | 1.195 | 1.128 | 1.742 |
| 19 | 1.017 | 1.468 | 1.232 | 1.271 | 1.189 | 1.840 |
| 20 | 1.075 | 1.550 | 1.298 | 1.349 | 1.251 | 1.939 |
| 21 | 1.134 | 1.631 | 1.364 | 1.429 | 1.312 | 2.038 |
| 22 | 1.194 | 1.714 | 1.431 | 1.512 | 1.373 | 2.137 |
| 23 | 1.254 | 1.798 | 1.497 | 1.598 | 1.433 | 2.237 |
| 24 | 1.316 | 1.884 | 1.564 | 1.686 | 1.493 | 2.338 |

Average Response Times (including service)

| N | i= 1 | i= 2 | i= 3 | i= 4 | i= 5 | i= 6 |
|---|---|---|---|---|---|---|
| 1 | 2.000 | 2.500 | 3.250 | 3.500 | 4.250 | 4.500 |
| 2 | 2.000 | 2.500 | 3.250 | 3.500 | 4.250 | 4.500 |
| 3 | 2.000 | 2.500 | 3.250 | 3.500 | 4.250 | 4.500 |
| 4 | 2.000 | 2.500 | 3.250 | 3.500 | 4.250 | 4.500 |
| 5 | 2.000 | 2.500 | 3.250 | 3.501 | 4.250 | 4.500 |
| 6 | 2.001 | 2.500 | 3.250 | 3.503 | 4.250 | 4.500 |
| 7 | 2.002 | 2.500 | 3.250 | 3.506 | 4.250 | 4.500 |
| 8 | 2.003 | 2.501 | 3.250 | 3.510 | 4.250 | 4.500 |
| 9 | 2.005 | 2.501 | 3.251 | 3.515 | 4.250 | 4.500 |
| 10 | 2.007 | 2.502 | 3.252 | 3.523 | 4.250 | 4.501 |
| 11 | 2.010 | 2.504 | 3.252 | 3.532 | 4.250 | 4.501 |
| 12 | 2.014 | 2.505 | 3.254 | 3.543 | 4.250 | 4.502 |
| 13 | 2.018 | 2.508 | 3.255 | 3.557 | 4.251 | 4.504 |
| 14 | 2.023 | 2.511 | 3.258 | 3.573 | 4.251 | 4.506 |
| 15 | 2.029 | 2.515 | 3.260 | 3.591 | 4.251 | 4.509 |
| 16 | 2.036 | 2.520 | 3.264 | 3.613 | 4.252 | 4.513 |
| 17 | 2.044 | 2.526 | 3.268 | 3.638 | 4.252 | 4.518 |
| 18 | 2.053 | 2.533 | 3.273 | 3.666 | 4.253 | 4.524 |
| 19 | 2.063 | 2.541 | 3.279 | 3.697 | 4.254 | 4.532 |
| 20 | 2.074 | 2.550 | 3.285 | 3.733 | 4.256 | 4.541 |
| 21 | 2.087 | 2.561 | 3.293 | 3.772 | 4.257 | 4.552 |
| 22 | 2.101 | 2.573 | 3.302 | 3.816 | 4.259 | 4.566 |
| 23 | 2.115 | 2.587 | 3.312 | 3.864 | 4.262 | 4.581 |
| 24 | 2.132 | 2.603 | 3.323 | 3.917 | 4.264 | 4.599 |

Average Utilizations

| N | i= 1 | i= 2 | i= 3 | i= 4 | i= 5 | i= 6 |
|---|---|---|---|---|---|---|
| 1 | 0.052 | 0.076 | 0.065 | 0.064 | 0.063 | 0.097 |
| 2 | 0.104 | 0.153 | 0.129 | 0.127 | 0.126 | 0.193 |
| 3 | 0.157 | 0.229 | 0.194 | 0.191 | 0.189 | 0.290 |
| 4 | 0.209 | 0.306 | 0.259 | 0.255 | 0.252 | 0.387 |
| 5 | 0.261 | 0.382 | 0.323 | 0.318 | 0.314 | 0.484 |
| 6 | 0.313 | 0.459 | 0.388 | 0.382 | 0.377 | 0.580 |
| 7 | 0.365 | 0.535 | 0.453 | 0.446 | 0.440 | 0.677 |
| 8 | 0.417 | 0.612 | 0.517 | 0.509 | 0.503 | 0.774 |
| 9 | 0.470 | 0.688 | 0.582 | 0.573 | 0.566 | 0.870 |
| 10 | 0.522 | 0.765 | 0.646 | 0.637 | 0.629 | 0.967 |
| 11 | 0.574 | 0.841 | 0.711 | 0.700 | 0.691 | 1.063 |
| 12 | 0.626 | 0.917 | 0.775 | 0.764 | 0.754 | 1.160 |
| 13 | 0.678 | 0.993 | 0.840 | 0.827 | 0.816 | 1.256 |
| 14 | 0.729 | 1.069 | 0.904 | 0.890 | 0.879 | 1.352 |
| 15 | 0.781 | 1.145 | 0.968 | 0.953 | 0.941 | 1.448 |
| 16 | 0.833 | 1.220 | 1.032 | 1.016 | 1.003 | 1.543 |
| 17 | 0.884 | 1.295 | 1.095 | 1.079 | 1.065 | 1.638 |
| 18 | 0.935 | 1.370 | 1.159 | 1.141 | 1.127 | 1.733 |
| 19 | 0.986 | 1.445 | 1.222 | 1.203 | 1.188 | 1.827 |
| 20 | 1.037 | 1.519 | 1.284 | 1.265 | 1.249 | 1.921 |
| 21 | 1.087 | 1.593 | 1.346 | 1.326 | 1.310 | 2.014 |
| 22 | 1.137 | 1.666 | 1.408 | 1.387 | 1.370 | 2.107 |
| 23 | 1.186 | 1.738 | 1.469 | 1.447 | 1.429 | 2.198 |
| 24 | 1.235 | 1.809 | 1.530 | 1.507 | 1.488 | 2.288 |

# *Bibliography*


1. Abell, John B. *The Sortie Generation Model System, Volume I: Executive Summary.* Contract MDA903-80-C-0554, Logistics Management Institute, September 1981 (AD-A110897).

2. Abell, John B. *The Sortie Generation Model System, Volume III: Sortie Generation Model Analyst's Manual.* Contract MDA903-81-C-0166, Logistics Management Institute, September 1981 (AD-A110897).

3. Bennett, G. Kemble and Brian J. Melloy. "Applying Queuing Theory Helps Minimize Waiting Time and Costs of Available Resources," *Industrial Engineering*, *16*:86–92 (July 1984).

4. Boling, Ronald W. and Frederick S. Hillier. *A Fleet Maintenance System Design Model.* Technical Report 214, Stanford, CA: Stanford University, December 1984.

5. Boyle, Edward. *LCOM Explained.* Technical Report AFHRL-TP-90-58, Wright-Patterson AFB, OH: Logistics and Human Factors Division, Human Resources Laboratory, July 1990.

6. Boyle, Edward and others. *Small Unit Maintenance Manpower Analyses (SUMMA).* Technical Report AFHRL-TR-89-26, Wright-Patterson AFB, OH: Logistics and Human Factors Division, March 1990.

7. Bulgak, A. A., et al. "Research on the Dynamics and Design Optimization of Automatic Assembly Systems." *Proceedings of the 15th Conference on Production Research and Technology.* 275–283. SME, 1989.

8. Byrd, Jr, Jack. "The Value of Queueing Theory," *Interfaces*, *8*:22–26 (May 1978).

9. Cronk, Richard and Alan J. Wallace. *LCOM User Manual, Version 93.C.* Technical Report, Wright-Patterson AFB, OH: Engineering Division, Aeronautical Systems Center, October 1993.

10. Department of the Air Force. *Logistics Composite Modeling.* AFR 25-7. Washington: HQ USAF, 1987.

11. Dietz, Dennis C. and Matthew Rosenshine. "Optimal Specialization of Maintenance Manpower." unpublished, 1994.

12. Drake, III, William F. and Raymond C. Reiss. *LCOM II Simulation Software Users Reference Guide.* Technical Report AFMSMET Report 78-5.1, Air Force Management Engineering Agency, October 1979.

13. Emerson, Donald E. *An Introduction to the TSAR Simulation Program.* Technical Report R-2584-AF, The Rand Corporation, February 1982.

14. Fisher, R. R., et al. *The Logistics Composite Model: An Overall View*. Technical Report RM-5544-PR, The Rand Corporation, May 1968.

15. Gelenbe, E. and G. Pujolle. *Introduction to Queueing Networks*. New York: John Wiley and Sons, 1987.

16. Hillier, Frederick S. and Gerald J. Lieberman. *Operations Research* (Fifth Edition). San Francisco: Holden-Day, Inc., 1992.

17. Kamath, M. and J. L. Sanders. "Modeling Operator/Workstation Interference in Asynchronous Automatic Assembly Systems," *Discrete Event Dynamic Systems: Theory and Applications*, *1*:93–124 (1991).

18. Kant, K. *Introduction to Computer System Performance Evaluation*. New York: McGraw-Hill Book Company, 1992.

19. Lavenberg, S. S. and M. Reiser. "Stationary Probabilities at Arrival Instants for Closed Queueing Networks with Multiple Types of Customers," *Journal of Applied Probability*, *17*:1048–1061 (1980).

20. Law, Averill M. and W. David Keiton. *Simulation Modeling and Analyses* (Second Edition). New York: McGraw-Hill Book Company, 1982.

21. Leung, Ying-Tat and Rajan Suri. "Performance Evaluation of Discrete Manufacturing Systems," *IEEE Control Systems Magazine*, *10*:77–86 (June 1990).

22. Miller, L. W., et al. *Dyna-Sim: A Nonstationary Queuing Simulation with Application to the Automated Test Equipment Problem*. Technical Report N-2087-AF, The Rand Corporation, July 1984.

23. Noble, David R. *Comparison of the TSAR model to the LCOM Model*. MS thesis, School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, September 1986.

24. Nolte, Jr, L. H. *Survey of Air Force Logistics Capability Assessment Concepts-Definitions-Techniques*. Technical Report AFLMC-781029-1, Gunter AFB, AL: Air Force Logistics Management Center, 1980.

25. Presutti, Jr, Victor, et al. *Maintenance Manpower, Spares Costs, and Repair Costs Impacts of Going to Two Levels of Maintenance for the B52H and the KC135*. Technical Report 89-213, Wright-Patterson AFB, OH: Concept Development Division, Headquarters, Air Force Logistics Command, December 1989.

26. Rao, P. Chandrasekhar and Rajan Suri. "Analytical Models for Closed Systems - Part II." unpublished, September 1993.

27. Richards, Jr, Eugene R. *Building and Operating the Logistics Composite Model (LCOM) for New Weapon Systems, Part A*. Technical Report ASD-TR-82-5033, Wright-Patterson AFB, OH: Engineering Specialities Division, Aeronautical Systems Division, February 1983.

28. Suri, Rajan, et al. "Performance Evaluation of Production Networks." *Handbooks in OR & MS 4*, edited by S.C. Graves and others, Elsevier Science Publishers, 1993.

29. Wallace, Alan J., LCOM Simulation Software Engineer. Personal Interview. ASC/ENNSC, Wright-Patterson AFB OH, 17 November 1993.

30. Wilson, Michael G. and others. *Optimizing Aircraft Maintenance Task/Specialty Assignments*. Technical Report AFHRL-TP-87-46, Wright-Patterson AFB, OH: Logistics and Human Factors Division, May 1988.

## *Vita*

Captain Richard C. Jenkins was born 8 May 1959 at Camp Le June, North Carolina. He graduated from high school in Hyattsville, Maryland in 1977 and enlisted in the United States Air Force in January 1978. Captain Jenkins was selected for the Airman's Education and Commissioning Program in 1987. In 1989, he graduated from New Mexico State University, Las Cruces, New Mexico, with the degree of Bachelor of Science in Mathematics. Captain Jenkins attended the United States Air Force Officer Training School and was commissioned in September, 1989. He was assigned to the Ballistic Missile Organization, Norton AFB, California until entering the School of Engineering, Air Force Institute of Technology in August, 1992.

Permanent address: 3927 Nicholson St
Hyattsville, Maryland
20782